Project Report: Asymptote Code Generation with Gemma 3

1. Project Goal:
To investigate the capabilities of Google's Gemma 3 language models for generating Asymptote vector graphics code. This involves creating a specialized dataset from Asymptote examples and setting up a framework for model fine-tuning and inference.
2. Methodology & Key Components:

- Dataset Creation (makeDataset.py):
    - A Python script was developed to automate the creation of a paired dataset. It processes .asy (Asymptote code) files from a designated source directory (e.g., asymptote-exemples/, which can be populated with any collection of Asymptote files).
    - The script compiles valid 2D Asymptote code, rendering PNG images which are stored in asy_images/.
    - The original Asymptote code is then paired with its rendered image, forming the asymp_dataset.json. This dataset is foundational for training models to understand the relationship between Asymptote code and its visual output.

- Model & Framework:
    - The project leverages Google's Gemma 3 series models (initially tested with unsloth/gemma-3-4b-it) via Unsloth's library, which facilitates efficient loading (e.g., 4-bit quantization) and LoRA-based fine-tuning.

- Model Interaction & Training Setup:
    - Inference scripts (Inference.py, Inference.ipynb) demonstrate loading Gemma 3 models and an interactive loop, showing a setup geared towards multimodal (image and text) input.
    - gemma3Training.py is the initial script for fine-tuning. While requiring further integration with the custom dataset for multimodal tasks, it establishes the Unsloth-based training environment.

3. Current Status & Outcomes:

- Successfully created a custom dataset (asymp_dataset.json and asymp_images/) linking Asymptote code with rendered visual outputs.

- Established a workflow for loading and interacting with Gemma 3 models (specifically, the 4B-IT variant was used in initial tests).

- Prepared the foundational scripts and dataset for future multimodal fine-tuning, aiming to enable the model to generate Asymptote code from visual or textual descriptions of diagrams.

4. Future Work & Potential Improvements:

- Scale Up Model: Transition from the currently used gemma-3-4b-it to a larger, more capable model like unsloth/gemma-3-27b-it. This would likely yield significant improvements in code generation quality and complexity, though it would necessitate more powerful GPU resources (e.g., rented GPU access).

- Increase Training Data & Steps:
    - Expand the asymp_dataset.json by incorporating more diverse and complex Asymptote examples.
    - Conduct more extensive fine-tuning runs with increased training steps/epochs on the chosen model.

- Full Multimodal Fine-tuning:
    - Adapt and complete the gemma3Training.py script to fully support multimodal fine-tuning using the asymp_dataset.json. This involves integrating image preprocessing an_d ensuring the training loop effectively learns from both image an_d code modalities.

- Refine Prompting & Evaluation:
    - Experiment with different prompting strategies to optimize the model's code generation based on visual inputs or detailed text descriptions.
    - Implement robust evaluation metrics to precisely measure the quality, correctness, and functionality of the Asymptote code generated by the fine-tuned model.

Conclusion:
The project has successfully established the core components for an Asymptote code generation system, including a custom dataset builder and an initial model interaction framework. The clear path forward involves scaling up the model, enriching the training data, and fully implementing multimodal fine-tuning to achieve the goal of proficient AI-driven Asymptote code generation.