**The Secrets of Pascal’s Triangle**

**A Visual Journey with MANIM**

**Submitted by:**

* Chintala Sejal - 2024021878
* P.G. Yuktha Reddy - 2024024788
* Yadla Sai Sushma - 2024165826
* Baswa Vinaya Sangeeta Lahari - 2024169782

**Abstract**

This project explores Pascal’s Triangle through a combination of visual and conceptual analysis using MANIM, an advanced animation library tailored for mathematical demonstrations. The initiative aims to enhance comprehension, particularly for learners who struggle with abstract concepts, by leveraging AI-generated MANIM code to dynamically illustrate the triangle’s construction, properties, and intricate patterns. These include symmetry, binomial coefficients, connections to the Fibonacci sequence, and fractals. The outcome is a **cinematic educational video** designed to transform complex mathematical ideas into intuitive and engaging visual experiences.

**Introduction**

Pascal's Triangle is a fundamental mathematical structure with applications spanning combinatorics, algebra, and number theory. Despite its significance, grasping its intricate patterns and relationships can be challenging for many learners. This project addresses these challenges by utilizing visual representations to demystify the triangle’s properties. By animating its construction and showcasing its diverse applications, the project seeks to make Pascal’s Triangle more accessible and engaging, fostering deeper understanding among learners of varying skill levels.

**Project Objective**

The main objectives of this project are:

* To visually represent Pascal’s Triangle and its mathematical patterns.
* To utilize the MANIM library to create a clear and structured educational animation.
* To employ AI models in generating optimized, functional code for MANIM.
* To make complex concepts accessible to a wider audience, particularly slow learners.
* To integrate creative and technical skills into a collaborative academic project.

**About MANIM**

MANIM is an **open-source Python library** by Grant Sanderson (3Blue1Brown) renowned for its capability to create precise and visually compelling mathematical animations. Its robust features make it an ideal tool for illustrating complex concepts in an educational context. In this project, MANIM was employed to animate various aspects of Pascal’s Triangle, including its formation and notable patterns. By harnessing MANIM’s capabilities, the project delivers clear and impactful visualizations that enhance learning outcomes and simplify abstract mathematical ideas.

**Project Description**

The project consists of a sequence of animated scenes, each designed to highlight distinct aspects of Pascal's Triangle. These animations delve into the triangle’s construction, symmetry, and its role in binomial expansions. Additionally, the project explores fascinating patterns embedded within Pascal’s Triangle, such as its connection to the Fibonacci sequence and the formation of the Sierpinski triangle. By presenting these concepts through visually engaging and intuitive animations, the project aims to make complex mathematical ideas more accessible and captivating for learners.

P**roject Link:** [**https://youtu.be/uOaWRVniQ68**](https://youtu.be/uOaWRVniQ68)

**Overview of Pascal's Triangle**

Pascal’s Triangle is a triangular array of binomial coefficients. Each number in the triangle is the sum of the two directly above it. Beyond this fundamental rule, the triangle contains a variety of patterns, such as:

* **Symmetry** about the vertical axis.
* **Powers of 2** across rows.
* **Fibonacci numbers** embedded in diagonal sequences.
* **Triangular numbers**, **tetrahedral numbers**, and other figurate number patterns.
* **Binomial expansions** corresponding to row entries.

These patterns were illustrated step-by-step in the animation, with appropriate labels and highlights to ensure clarity.

**Code and Features Used**

The project is structured into multiple scenes, each implemented as a separate MANIM class. Key features and functions used include:

**1. Scene Classes**

Every animation scene in MANIM is defined using a class that inherits from Scene. The animation logic is written inside the construct() method.

class PascalIntro(Scene):

def construct(self):

...

Each major concept in our video — triangle formation, Fibonacci paths, and Sierpinski patterns — is encapsulated in its own class for clarity and modularity.

**2. Basic Visual Elements**

We used MANIM’s rich object library to render visuals:

* Text() – For regular English text
* MathTex() – For mathematical LaTeX expressions
* Dot(), Line(), Arrow() – For geometric construction
* SurroundingRectangle() – To highlight grouped content

self.play(Write(MathTex("a^2 + b^2 = c^2")))

This helped us merge visuals with math-based storytelling.

**3. Animation Functions**

To bring static visuals to life, we used MANIM’s powerful animation tools:

* self.play() – Triggers animations
* Common animations: Write(), FadeIn(), FadeOut(), Create(), Transform()
* add() and remove() – Instant object display/removal
* wait(n) – Adds a pause to absorb content

self.play(FadeIn(title), run\_time=2)

Timing and transitions were planned to maintain engagement and improve retention.

**4. Positioning and Styling**

Precise layout and coloring made our visuals cleaner and more readable:

.move\_to() – Positions object absolutely

.shift(UP + LEFT) – Relative positioning

.scale(0.7) – Resizes objects

.set\_color(YELLOW) – Changes color

.next\_to() – Places objects beside others

text.scale(0.8).next\_to(title, DOWN)

These tweaks helped organize the triangle visually and logically.

**5. Loops and Dynamic Construction**

The triangle and many patterns were generated dynamically using nested Python loops with MANIM rendering:

for i in range(n):

for j in range(i + 1):

value = comb(i, j)

...

This allowed us to auto-calculate positions, values, and relationships efficiently — essential for building higher-order triangles and highlighting evolving patterns.

**6. Grouping with VGroup**

We used VGroup to group multiple objects, align them, and animate them together:

group = VGroup(obj1, obj2, obj3)

group.arrange(DOWN)

self.play(FadeIn(group))

This reduced code clutter and improved animation synchronization across similar elements.

**7. Modulo Pattern for Sierpinski Triangle**

One of the highlights was visualizing **Sierpinski’s Triangle** using modulo logic:

if value % 2 == 1:

dot.set\_color(BLUE)

else:

dot.set\_opacity(0)

By plotting odd values in black and hiding even ones, the classic Sierpinski pattern emerged naturally from Pascal’s triangle — a visual "aha!" moment for viewers.

**Use of Artificial Intelligence Tools**

We employed advanced AI models to assist in the development of this project:

* **ChatGPT**: Assisted in writing functions, debugging errors, and optimizing code logic.
* **DeepSeek**: Generated alternative approaches and improved the readability of animations.
* **Claude**: Helped structure animations and provided visual storytelling strategies.

**Highlights of the Video**

The video is segmented into distinct parts, each illustrating a unique property or pattern:

* **Introduction**: Sets the stage for the exploration of Pascal's Triangle.
* **Triangle Formation**: Demonstrates the step-by-step construction of the triangle.
* **Symmetry**: Highlights the mirror symmetry present in the triangle.
* **Binomial Expansion**: Connects the triangle's rows to coefficients in binomial expansions.
* **Fibonacci Sequence**: Reveals the presence of Fibonacci numbers along certain diagonals.
* **Power of 11**: Shows how the triangle's rows correspond to powers of 11.
* **Sierpinski Triangle**: Visualizes the fractal pattern emerging from coloring odd and even numbers.
* **Finale**: Concludes with a summary and appreciation of the triangle's beauty.

**Challenges and Learnings**

During the development of this project, we encountered several challenges:

* Understanding the MANIM library and its syntax.
* Visualizing patterns clearly without overcrowding the screen.
* Coordinating group contributions remotely.

However, through iterative development, prompt engineering, and peer collaboration, we overcame these challenges and gained:

* Practical experience in Python and MANIM scripting.
* Confidence in working with AI-assisted tools.
* Enhanced problem-solving and design skills.

**Future Scope**

Potential extensions of the project include:

* Developing interactive elements to allow users to explore the triangle dynamically.
* Adding explanatory voiceovers to guide viewers through the concepts.
* Publishing the video on educational platforms to reach a wider audience.

**Conclusion**

“The Secrets of Pascal’s Triangle” exemplifies how technology, mathematics, and pedagogy can be integrated to create effective learning resources. Using MANIM and AI-assisted development, we were able to present a classical mathematical concept in a modern, engaging, and accessible format. This project not only deepened our understanding of the subject but also provided us with valuable experience in collaborative development and educational design.

**References**

* Manim Community Developers. (n.d.). *Manim Community Edition Documentation*. Retrieved April 11, 2025, from <https://docs.manim.community/en/stable/>
* Math is Fun. (n.d.). *Pascal's Triangle*. Retrieved April 11, 2025, from <https://www.mathsisfun.com/pascals-triangle.html>
* TED-Ed. (2017, June 26). *The mathematical secrets of Pascal’s triangle* on YouTube. <https://www.youtube.com/watch?v=0iMtlus-afo>
* Numberphile. (2013, September 3). *Pascal’s Triangle* on YouTube. <https://www.youtube.com/watch?v=0iMtlus-afo>
* OpenAI. (2025). *ChatGPT (April 2025 version)* [Large language model]. <https://chat.openai.com/>
* Anthropic. (2025). *Claude AI* [Large language model assistant]. <https://www.anthropic.com/>
* DeepSeek. (2025). *DeepSeek AI Assistant*. <https://deepseek.com/>

**Appendices**

The complete source code made for the development of the MANIM-based animation on Pascal’s Triangle is maintained in a dedicated **Notion workspace.** This repository includes all essential Python scripts, class functions, and AI-assisted code blocks required to generate the visual elements of the project.

**Access the code repository at:**  
<https://www.notion.so/Codes-of-The-Secrets-of-Pascal-s-Triangle-Project-1bd44e9f648a801297eae94e0e48fe35?pvs=4>

The final animated video titled *"The Secrets of Pascal’s Triangle" is below:*

**Watch the final project video at:**  
 <https://youtu.be/uOaWRVniQ68?si=Th_IE-yrrort0USX>