

# Animato: A Domain-Specific Language for Animation and Drawing

DANIEL SON, Harvard University, USA

Animato is a domain-specific language aimed to tackle the problem of tedious drawing for animation and shape rendering. While there exists tools such as p5.js [p5j 2024] and Penrose [Ye et al. 2020], this project mimics and extends certain aspects of existing tools. We design a domain-specific language that takes shape inputs with locations and outputs the drawings on a canvas as a .png file. Depending on the input, the output may be a singular or multiple image files where the sequence of image files can be input into an animation software such as Adobe Premiere Pro or TV Paint to create an animation.

## 1 INTRODUCTION

Art is prevalent in all shapes and forms, one being visual. In the field of animation, drawing is a tedious yet important task that 2-dimensional animators often have to utilize. Due to the tedious nature of hand-drawn animation, Animato aims to relieve animators of the need to draw each frame and develop animations using the domain-specific language.

### 1.1 Related Work

Previously, much work has been done in the p5.js [p5j 2024] community as well as the Penrose [Ye et al. 2020] CMU community to advance the code in art space. p5.js is an open-source JavaScript library that aims to make creative coding accessible to artists and coders alike. Similarly, Penrose provides a tool to allow users to create complex diagrams that would otherwise be hard to depict in a formal setting.

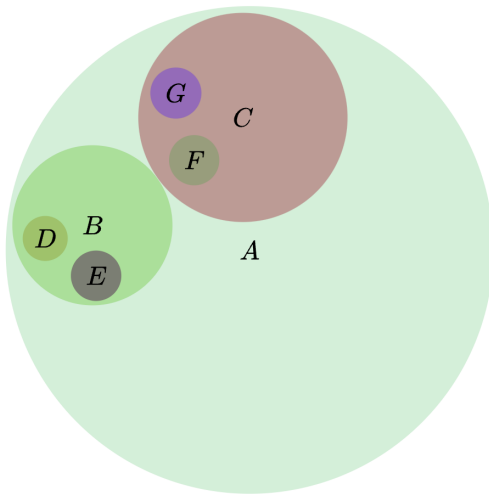


Fig. 1. Example diagram created in Penrose of sets

The overall goal of these languages and libraries is to provide an easier and more precise way to create drawings. In this project, we attempt to recreate features of p5.js and Penrose in Animato's

language and extend certain functionality to be more specialized. While Animato is not necessarily novel, it aims to be a more general and user-friendly language for animators who do not need to understand the backend.

## 2 FRAMEWORK

### 2.1 Primitives

In the current iteration of Animato, there are multiple primitives for shapes that are ready to be utilized. These include squares, circles, line segments, and some other harder to achieve common shapes. While circles and squares are predefined shapes, lines can be drawn and connected with other lines to create more complex shapes. In the future, it would be very beneficial to implement a primitive to create bezier curves. The addition of bezier curves would allow users to create shapes that are less rigid (e.g. a cloud shape).

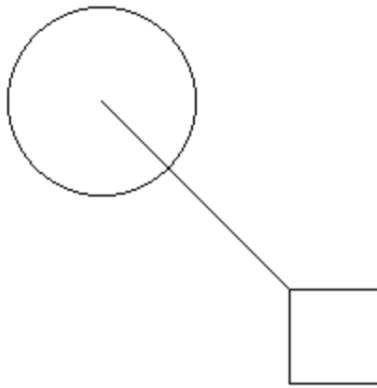


Fig. 2. Example of circle, square, line segment primitives on canvas

### 2.2 Implementation

Animato uses a Python backend with Pillow for drawing visualizations, z3 for constraint solving, and PLY/sexpr for parsing and interpreting the input. At the current state, running the `main.py` file with a valid `dsl_code` input will only produce one file. The file is a visual representation of the input given with precise locations and relationships.

## 3 EXAMPLE

Using these primitives, users are able to create custom shapes (functions) by defining a function as follows:

Listing 1. Sample custom function definition in Animato

```
dsl_code = """
(define star (cx 0) (cy 0) (radius 50) (
```

```

    (line (x1 (- cx radius))
          (y1 (- cy radius))
          (x2 (+ cx radius))
          (y2 (+ cy radius)))
    (line (x1 (- cx radius))
          (y1 (+ cy radius))
          (x2 (+ cx radius))
          (y2 (- cy radius)))
    (line (x1 cx) (y1 (- cy radius))
          (x2 cx) (y2 (+ cy radius)))
    (line (x1 (- cx radius)) (y1 cy)
          (x2 (+ cx radius)) (y2 cy))
  ))
  (star (cx 250) (cy 250) (radius 100))
  " " "

```

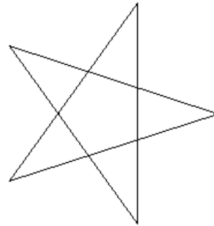


Fig. 3. Output image of `dsl_code` input for a star

The above `dsl_code` input produces the drawing shown in figure 1.

#### 4 LIMITATIONS AND FUTURE WORK

While the intent of this project is to create a domain-specific language to assist programmers and animators to easily draw animations, there is still work that needs to be done in order to flesh out this idea. Firstly, the aspect of custom-defined functions is quite limited and Animato needs more functionality that gives a more user-friendly experience. While the aim of Animato is to allow everyone to create animations without the need to look in the backend to see what it's doing, this is quite difficult to allow users to actually do whatever they want. In testing, it has been apparent that

actually creating custom functions is quite hard since the user needs a precise understanding of the canvas space. Further, without bezier curves, users are not able to create curved shapes making the scope of possible shapes limited. In future iterations of Animato, the main focus should be to create a more user-friendly experience and more functionality in the language itself. While not in the scope for this project, ideally users would be able to create a shape in the `dsl_code` input and have a live view of the canvas. Currently, the only way to test the drawings is by running `main.py` and checking the output `.png` file.

## 5 CONCLUSION

Overall, while the aim of Animato is to provide a more concise and specialized language for animators/developers to use, further work must be done to polish the domain-specific language. At the current state, users are able to create a canvas with some specified shape drawings, including custom shapes. Further work should be done to extend the usability and enhance existing features. This project acts as a foundation that can be extended for more utility.

## ACKNOWLEDGMENTS

This paper and project serves as my final project for CS 252, Advanced Programming Languages, at Harvard University. Special thanks to Professor Nada Amin, Raffi Sanna, and classmates in CS 252 for providing feedback and advice on the project in it's developmental stage.

## REFERENCES

2024. p5.js. <https://p5js.org/>
- Katherine Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. 2020. Penrose: from mathematical notation to beautiful diagrams. *ACM Trans. Graph.* 39, 4, Article 144 (Aug. 2020), 16 pages. <https://doi.org/10.1145/3386569.3392375>