DevForge Problem Statement: Text-Driven Educational Animation Video Generator

---

1. Problem Statement

Build a platform that converts **natural language descriptions** into **automatically generated educational animation videos**.
A user should be able to describe any academic concept—such as math steps, physics processes, algorithms, diagrams, graphs, vector operations, scientific explanations—and the system should automatically convert the description into:

- A structured set of animation instructions
- Rendered visual animations
- A final educational video explaining the concept

The system should support both simple and complex educational visuals depending on the team's capabilities.
Participants may choose **any internal animation engine or rendering method**, as long as the video is generated automatically from text input.

---

2. Background & Why It Is Needed

Creating high-quality educational animations (like those seen in math channels, physics explainers, algorithm videos) is extremely time-consuming and requires:

- Technical animation skills
- Programming knowledge
- Expertise with motion design tools
- Manual scene building

- A long rendering workflow

This puts animated learning content out of reach for most students, teachers, and creators.

A platform that takes **text → visual explanation video** can revolutionize:

- Classroom education
- Self-learning platforms
- YouTube educational content
- Concept visualization
- Scientific communication
- Algorithm demonstration

Users can type:

“Show how a sine wave forms, label amplitude, show one period, and animate a moving point along it.”

And automatically receive a fully animated educational video.

---

3. Requirements

Core Requirements

- Accept natural-language descriptions of educational concepts.
- Convert them into a structured set of animation scenes.
- Automatically render visuals such as:
    - Geometric shapes
    - Mathematical formulas
    - Moving vectors

    - Graphs and plots
    - Step-by-step derivations
    - Algorithmic visualizations
    - Scientific diagrams
- Produce a final **educational animation video file** or **browser-based video preview**.
- Show the generated underlying animation logic (scene breakdown, timeline, instructions).
- Handle errors gracefully and show meaningful feedback.

Flexibility

Teams are free to choose:

- Any rendering engine
- Any graphics pipeline
- Any animation format
- Scene-based or timeline-based architecture
- Single or multiple scenes
- Any internal code-generation or rule-based mapping approach
- LLM-assisted or rule-based text interpretation

The focus is on the **text → educational animation pipeline**, not the specific engine.

---

4. What You Have to Build

Participants must deliver a working prototype that:

1. Takes text input

Examples:

- “Animate the Pythagoras theorem with a growing square.”
- “Show vector addition visually using arrows.”
- “Demonstrate bubble sort with colored bars.”
- “Plot a sine graph and highlight maxima/minima.”

2. Converts text into animation instructions

- Scene structure
- Object definitions
- Movements and transitions
- Timeline or sequence order

3. Renders the animation

- Generates a playable educational animation video
- OR a browser-based animation output

4. Displays outputs

- Final rendered animation
- Underlying animation instructions
- Scene breakdown
- Any generated intermediate code (optional)

5. Error handling

If generation or rendering fails, the system must explain what went wrong.

---

5. Example Pipeline Components (Reference Only)

Teams may choose any structure.

Text → Scene Breakdown

```
POST /generate/scenes
```

```
{ "description": "Show a triangle rotating and label its sides." }
```

Scene → Animation Instructions

```
POST /generate/instructions
```

```
{ "scene": "...parsed structure..." }
```

Render Animation

```
POST /render
```

```
{ "instructions": "...animation objects and actions..." }
```

Preview Video

```
GET /preview/{id}
```

These endpoints are **suggestive**, not mandatory. Teams may modify all components.

---

6. Evaluation Criteria

Round 1: Technical Qualifier (50 pts)

Focus: core functionality + reliable logic.

Text Parsing & Scene Generation (15 pts)

The system must convert descriptions into structured instructions.

Rendering Capability (10 pts)

Must demonstrate at least simple educational animations.

Scene Logic & Timeline Structure (10 pts)

Clear visual structure, correct sequencing.

Test Case Animation Outputs (10 pts)

Teams must pass predefined test descriptions such as:

- “Animate a circle expanding into a sphere.”
- “Visualize matrix multiplication step-by-step.”

Code Quality & Architecture (5 pts)

Teams must score ≥30 pts + test case compliance to enter Round 2.

Round 2: Final Demo & Judging (100 pts)

Focus: visual quality, completeness, user experience.

Quality of Generated Animations (30 pts)

Clarity, smoothness, usefulness for education.

Accuracy of Text-to-Visual Mapping (25 pts)

How well the system interprets complex input.

Engineering Depth & System Design (20 pts)

Rendering pipeline, animation logic, modular architecture.

User Interface & Experience (15 pts)

Ease of use, clarity, preview features.

Presentation & Storytelling (10 pts)

---

8. Notes

- Teams may use **any underlying animation system** or build their own.
- LLM-based code/instruction generation is allowed but not required.
- Final output must be a **rendered educational animation video** or **animating visual output**.
- OSC & AI/ML Club members must be given full code access (read + clone). Failure to provide access leads to disqualification.