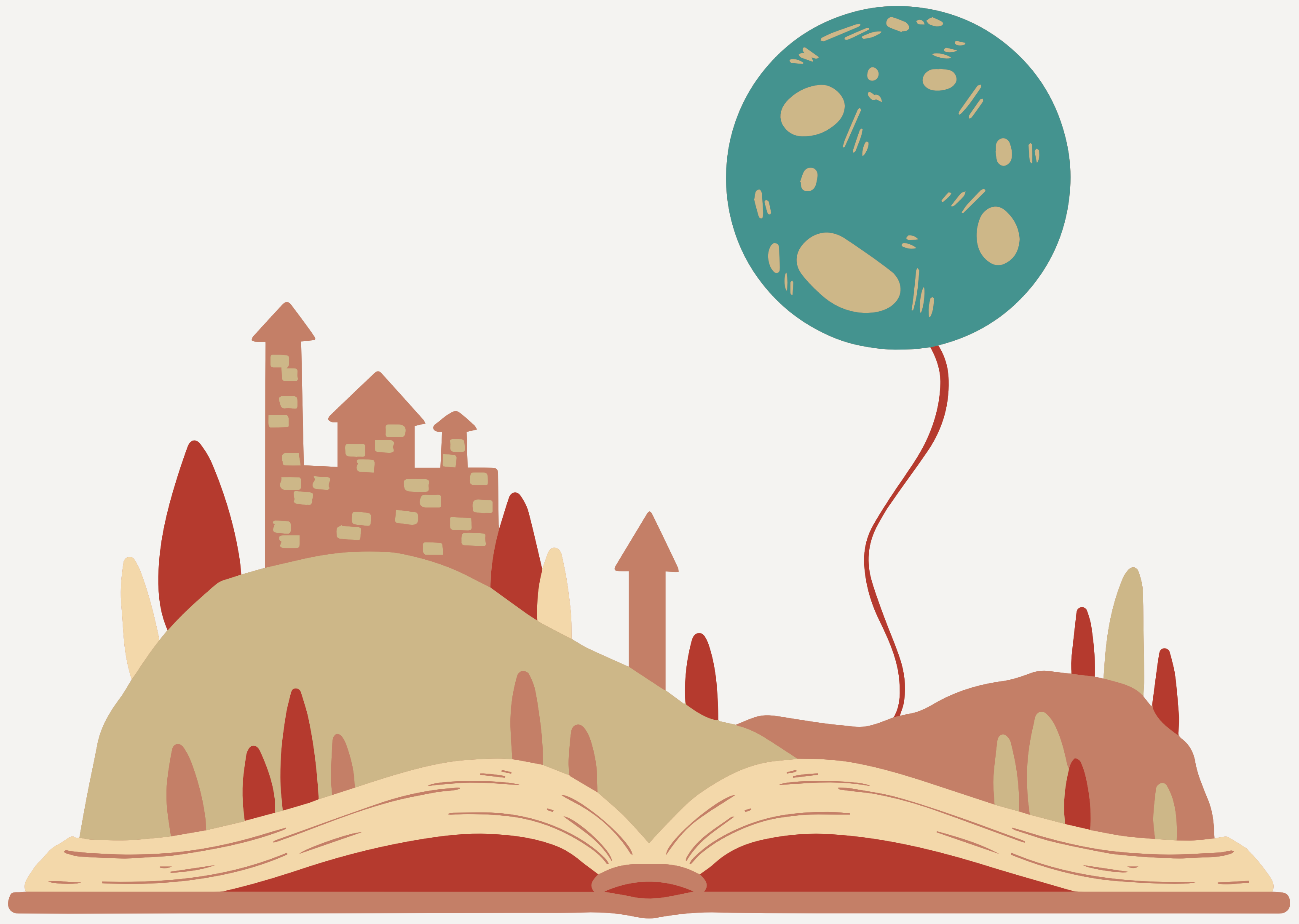
INTERACTIVE STORY TELLER

Laura Beltrán
Santiago Sánchez

TABLE OF CONTENTS

01. Introduction
02. Goal
03. Proposed Solution
04. Experiments
05. Expected Results
06. Development
07. Preliminary Results
08. Conclusions
09. References

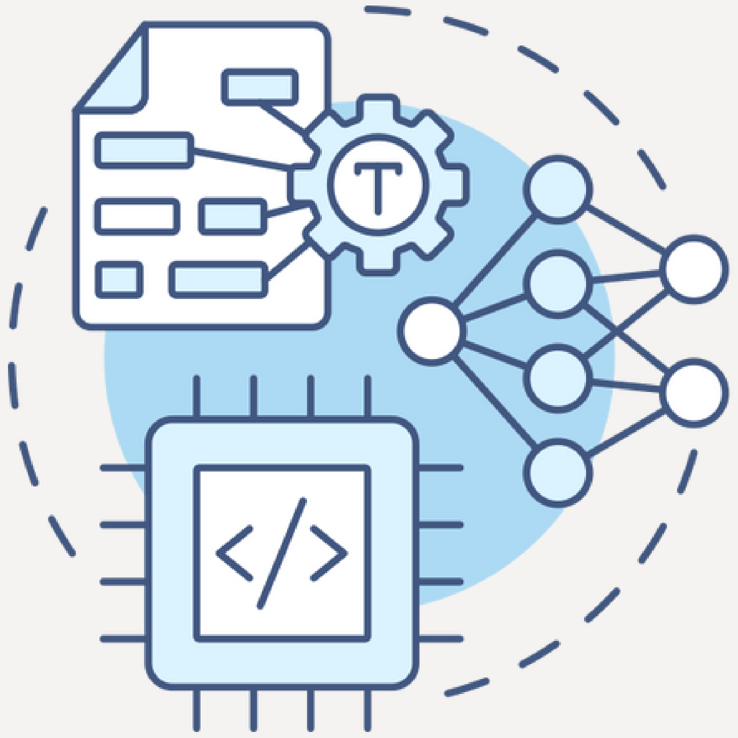
INTRODUCTION

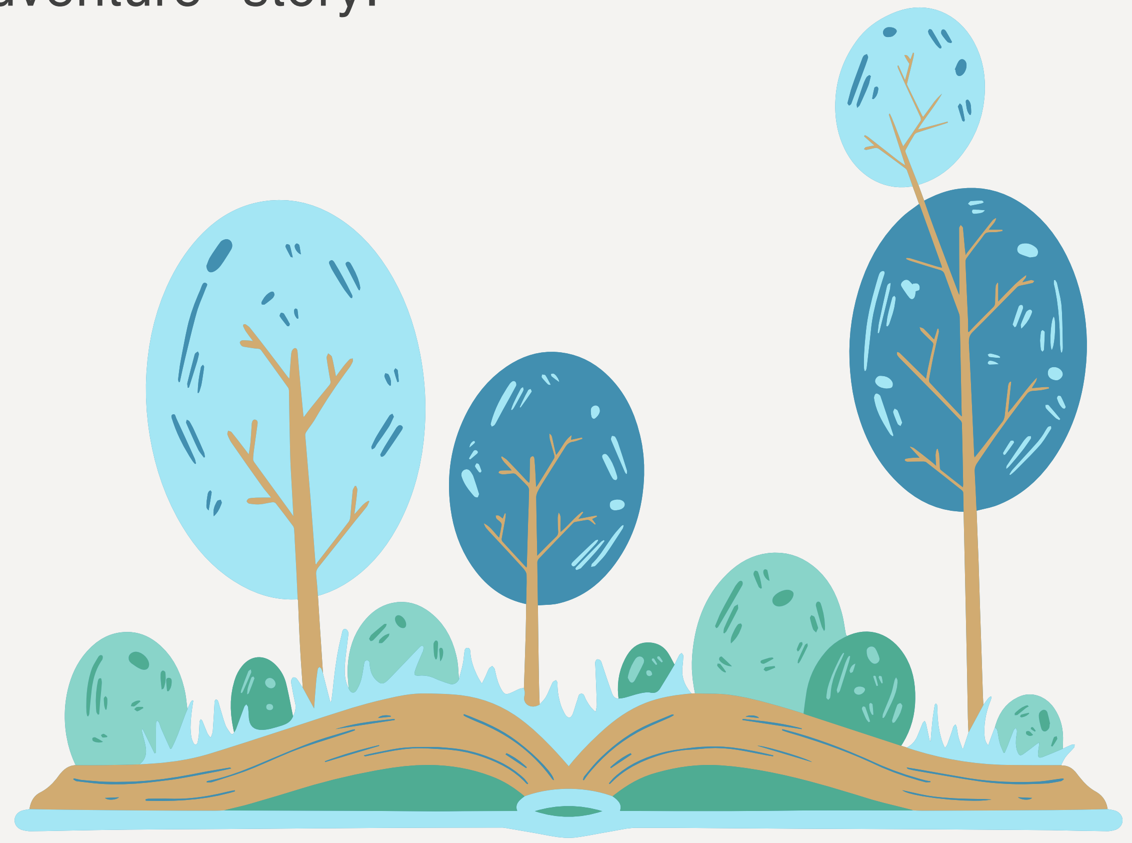


- **Traditional Compilers:** Transform programming languages into machine code.
- **Natural Language:** Applying computation theory to natural language is a complex challenge.
- **Educational Tool:** A compiler that converts structured natural language stories into interactive narratives.
- **Purpose:** Helps visualize abstract concepts such as finite-state machines and grammars.
- **Platform:** Browser-based tool for engaging and interactive learning.

GOAL

Create a compiler that converts structured natural language into an interactive "choose-your-own-adventure" story.





PROPPOSED SOLUTION

1

Story Structure:

- **Scenes:** Represented as FSM states.
- **Choices:** Represented as transitions between states.

2

Output:

- **HTML + JS:** Dynamic content with buttons to navigate through the story.

3

Implementation:

- **Language:** Python for parsing.
- **Parsing Method:** Regular expressions to interpret and process the input.

T E S T I N G

E X P E R I M E N T S

- Initial Testing: Using manually written stories that follow the defined structure.
- Focus Areas:
  a. Validate Scene Parsing
  b. Test Transitions Logic.
  c. Ensure Correct HTML Generation
- Input Structure:
  - Mirrors a context-free grammar.
  - Non-terminals: Represent scenes.
  - Production Rules: Define user decisions and transitions between scenes.

E X P E C T E D R E S U L T S

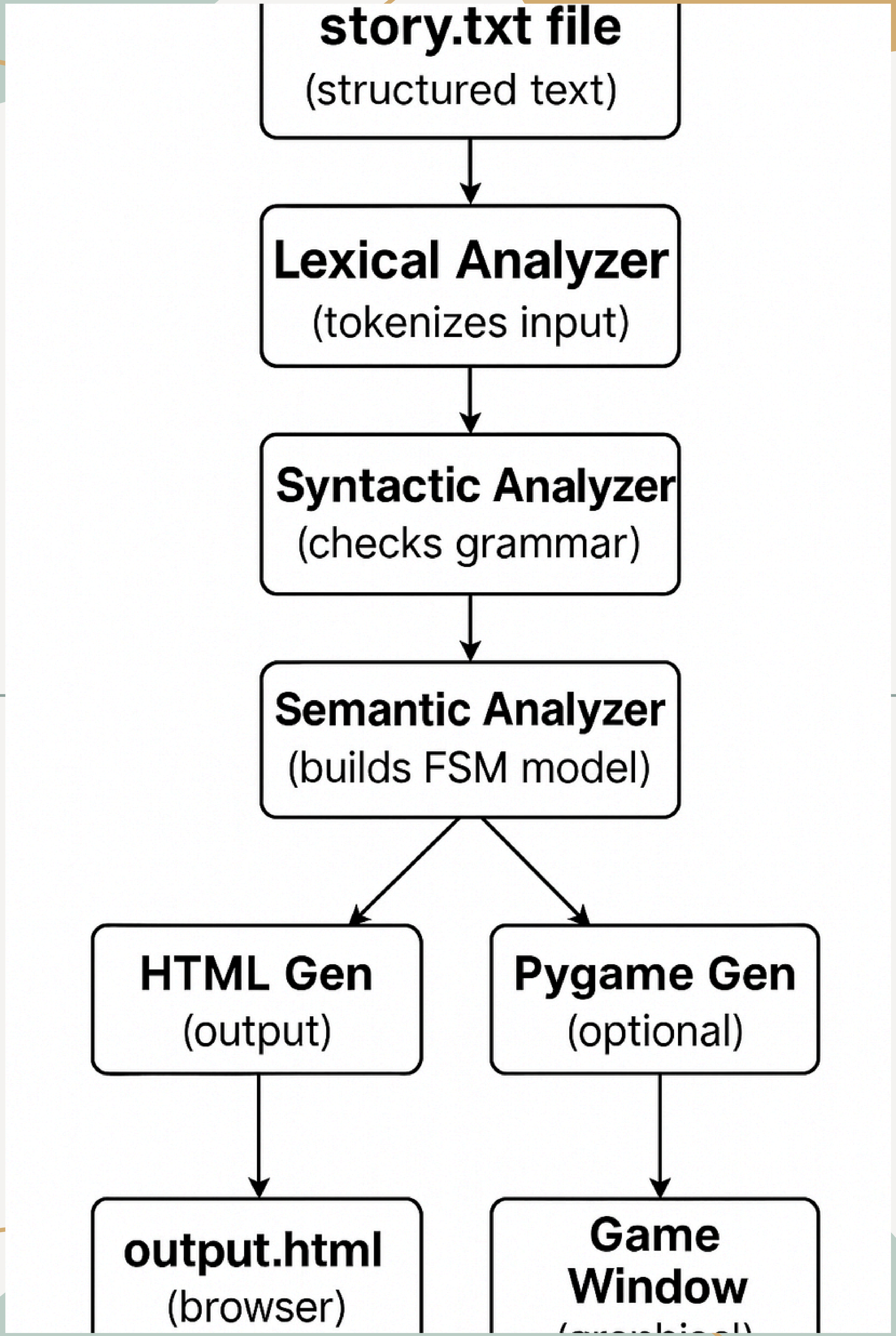
- Generate standalone HTML narratives
- Parse and validate simple branching stories
- Demonstrate FSM and CFG visually

CONCLUSIONS

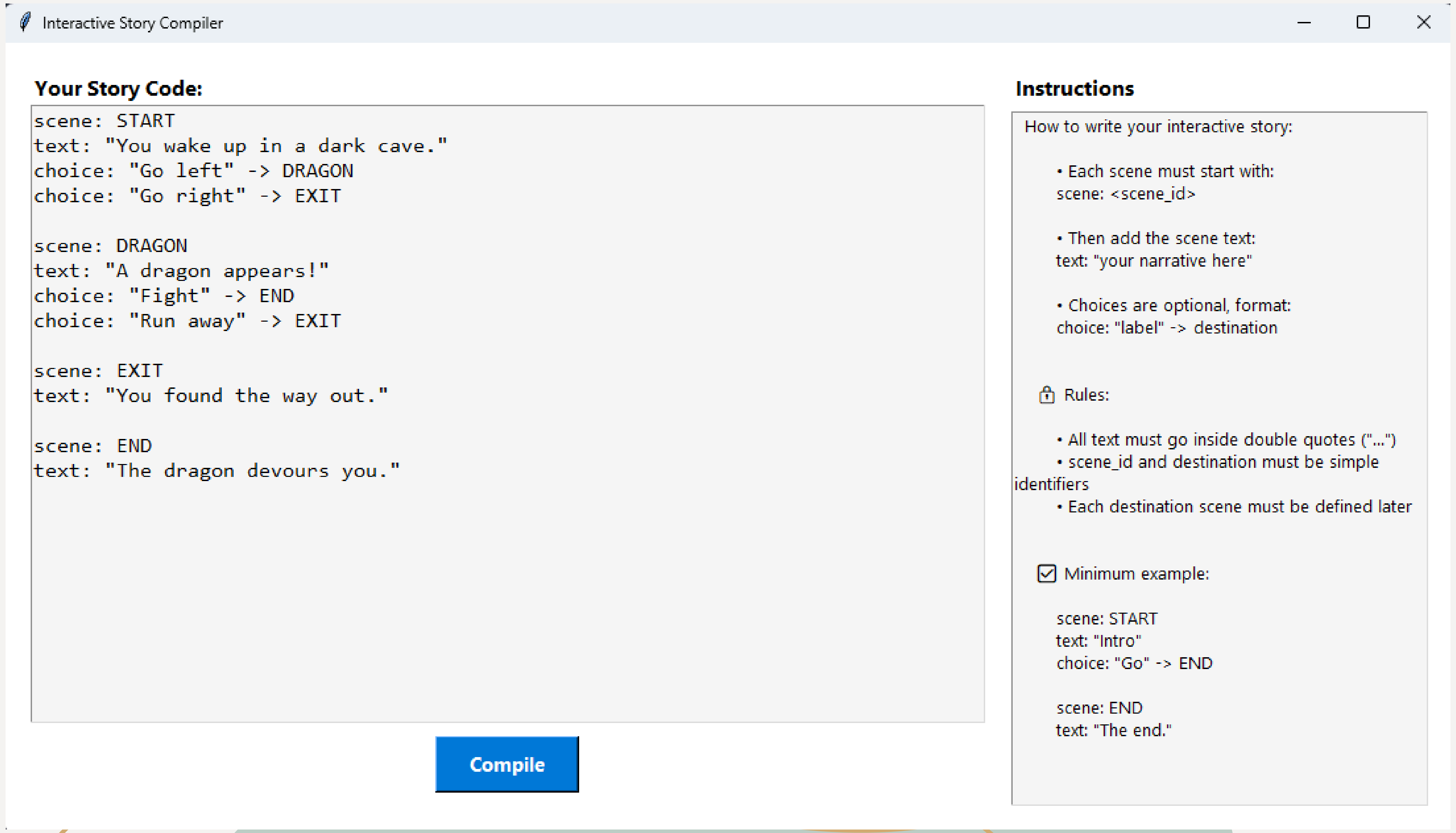
This compiler bridges the gap between theory and application. It reinforces computation concepts through creative output and encourages understanding of grammars, parsing, and machine logic in an interactive context.

DEVELOPMENT

PHASES


story.txt file
(structured text)
Lexical Analyzer
(tokenizes input)
Syntactic Analyzer
(checks grammar)
Semantic Analyzer
(builds FSM model)
HTML Gen
(output)
Pygame Gen
(optional)
output.html
(browser)
Game
Window

PRELIMINARY RESULTS



Your Story Code:

```
scene: START
text: "You wake up in a dark cave."
choice: "Go left" -> DRAGON
choice: "Go right" -> EXIT

scene: DRAGON
text: "A dragon appears!"
choice: "Fight" -> END
choice: "Run away" -> EXIT

scene: EXIT
text: "You found the way out."

scene: END
text: "The dragon devours you."
```

Instructions

How to write your interactive story:

- Each scene must start with:
  scene: <scene_id>
- Then add the scene text:
  text: "your narrative here"
- Choices are optional, format:
  choice: "label" -> destination

🔒 Rules:

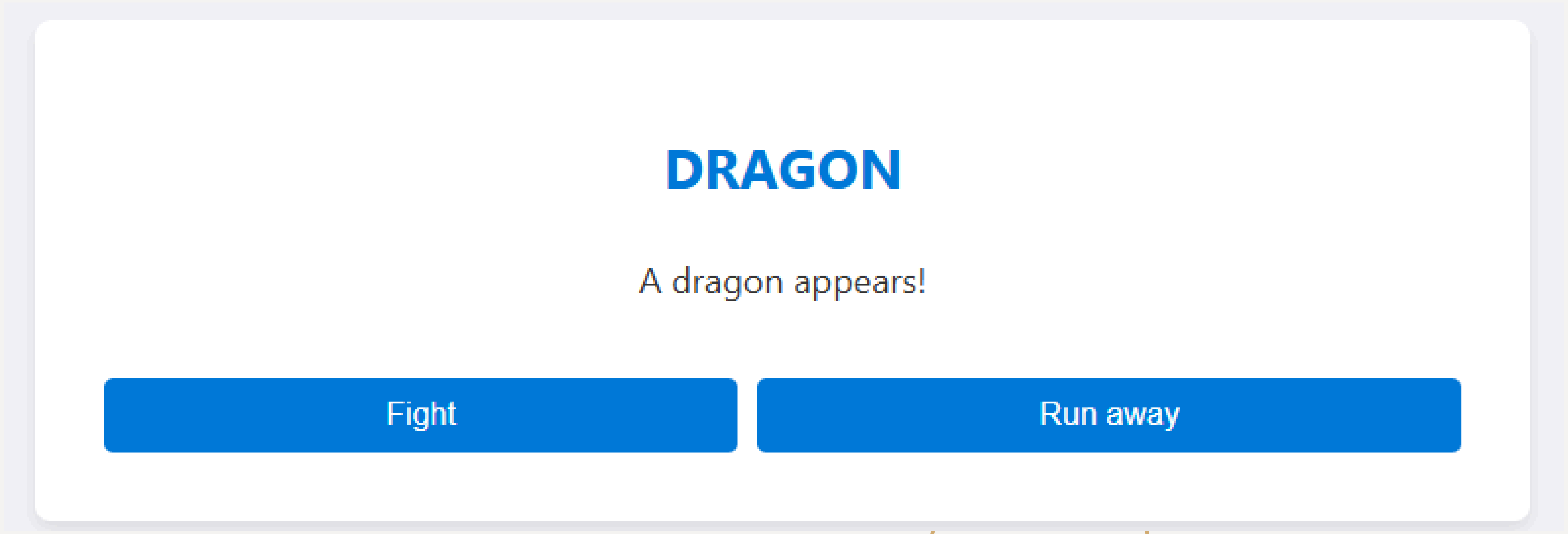
- All text must go inside double quotes ("...")
- scene_id and destination must be simple identifiers
- Each destination scene must be defined later

☑ Minimum example:

```
scene: START
text: "Intro"
choice: "Go" -> END

scene: END
text: "The end."
```

PRELIMINARY RESULTS


DRAGON

A dragon appears!

CONCLUSIONS

- Scalable
- Interactive

REFERENCES

- Twine. https://twinery.org
- Ink. https://www.inklestudios.com/ink
- Sierra, C. A. (2025). Slides from Computer Science III: Theory of Computation. Universidad Distrital.