Animations are ubiquitous
Animations convey complex topics, e.g., how to build a trie from strings, how TCP's three-way handshake works, or how mark & sweep garbage collection is performed.

The Pain Point
Existing tools (PowerPoint and Latex) are not designed for animations. Modifying an existing animation is hard, and software tools like source control are impractical.

What is THAPL?
A declarative domain-specific programming language inspired by theater plays, that creates slide-based animations.
- Enables the programmer to focus on behavior.
- Compiler handles details, no need to count slides.
- Expressive, flexible syntax.

Code example: Making a trie for “ramp” and “race”

**Act Trie:**
**Dramatis Personae:**
nodes {show, hide, connect, flash}:
  r, a, m, and p [below: "auto"]{c, and e (r)[leftof: "a"]).

**Action:**
r show.
a show meanwhile r connect to a.
m show meanwhile a connect to m.
p show meanwhile m connect to p.
r flash.
a flash.
c show meanwhile a connect to c.
e show meanwhile c connect to e.
nodes flash.

Object model
THAPL uses prototype inheritance (like JavaScript) for nouns and verbs so new objects can be based on the existing ones. Objects are defined using an object notation syntax (THAPLON). The 3 basic meta-types are organized as follows:

```
+ noun
  + verb list
  + properties
+ verb
  + modifier list
  + action
+ modifier
  + pattern clauses
  + context action
```

Features

**Declarative time-management** Compound sentences are constructed using time-management keywords. Sentences stretch to fit the alloted time.

**Meanwhile** makes actions occur in parallel—start and end at the same instant.

**English-like syntax** The basic construct that defines behavior is the sentence, which is comprised of nouns (on-screen objects), verbs (actions on those objects), and adverbs (modifiers on the actions).

Structure
Like a play, a program is made of sections. 
**Acts** divide animation into parts. 
**Dramatis Personae** lists nouns in an hierarchy. 
**Action** specifies act behavior.

Future directions

- Constraint based time-management for UX modeling.
- Stretchable calculations while waiting for other code, e.g., enhanced precision or logging.

*This research was supported in part by the Israel Science Foundation (ISF) under grant No. 1440/14.*