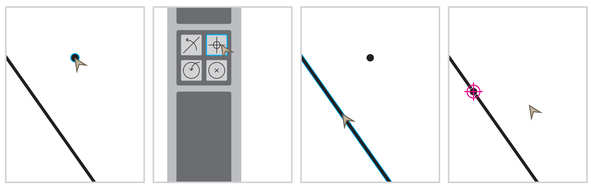
Construct
Programming with Geometry

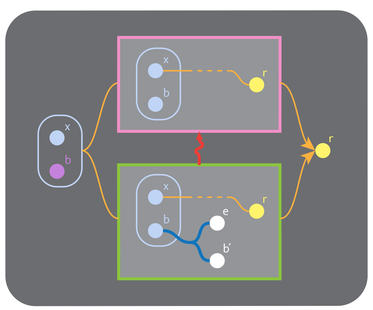
Sam Gruber Advisors: Brad Myers, Stephanie Murray

Geometric objects represent all data in Construct. There are three kinds of objects:
- **Literals** (points, lines and circles)
- **Measures** (angles and distances)
- **Aggregates** (sets and groups)

Definitions are used to specify objects through geometric relationships. They are applied to a single object, but may force recomputation of other objects in order to resolve to a valid program state. Every program in Construct is a definition which can be called from other programs.

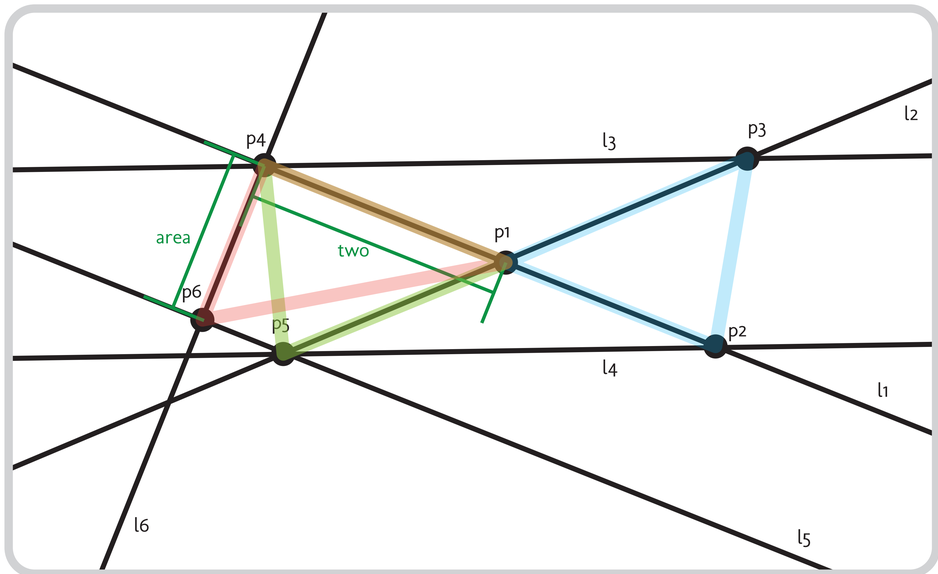


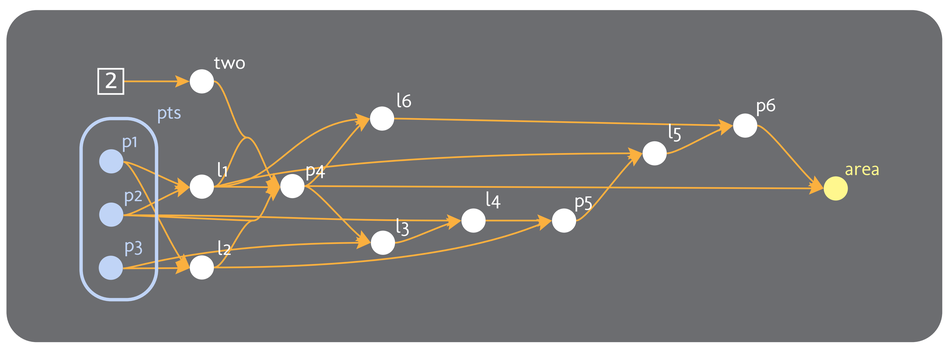
Idioms such as '**if**-**then**-**else**' below show common features in the terms of Construct.



If the set **b** is empty (false), then branch by **handling** the exception raised on attempting to extract an element from it.

Construct uses programming with examples in a geometric syntax to create programs with a geometric meaning as clearly as possible. **Construct aims to broaden computer programming to accept more contributions from visual and spatial thinkers**, making possible new understandings of computer science.





Above is a Construct program which uses elementary geometric techniques to find the area of the triangle highlighted in blue.

Visual professionals who wish to apply computing to their existing process are the target users of Construct. These users may not write code, but are looking to automate the generation of geometry from geometry.

Four goals shaped the design of Construct:
- **Clarity of program state.** Programs are shown both as an evaluation graph and as the state of the objects.
- **Native geometry concepts.** Objects and definitions mirror familiar ideas from pen and paper geometry.
- **Specification without inference.** Definitions of objects are always given as rules, rather than guessed from the programmer's demonstration.
- **A full range of programs.** Programmers access all of the capability of a Turing-complete language through geometry.

Future work remains to extend and implement the system:
- **Runtime Implementation** to allow programs to be run and tested.
- **Interface Implementation** and user testing to assess the effectiveness of geometric programming.
- **Graph Cut and Paste** for more efficient development of programs in Construct.
- **Higher Dimensions** and **Datatypes** to express complex program designs.