Sleek A Visual Scripting Language

Ryan Taus

Outline

- What is a Visual Scripting Language?
- About Sleek
- Interaction Design
- Compilers
- Challenges
- Future Work

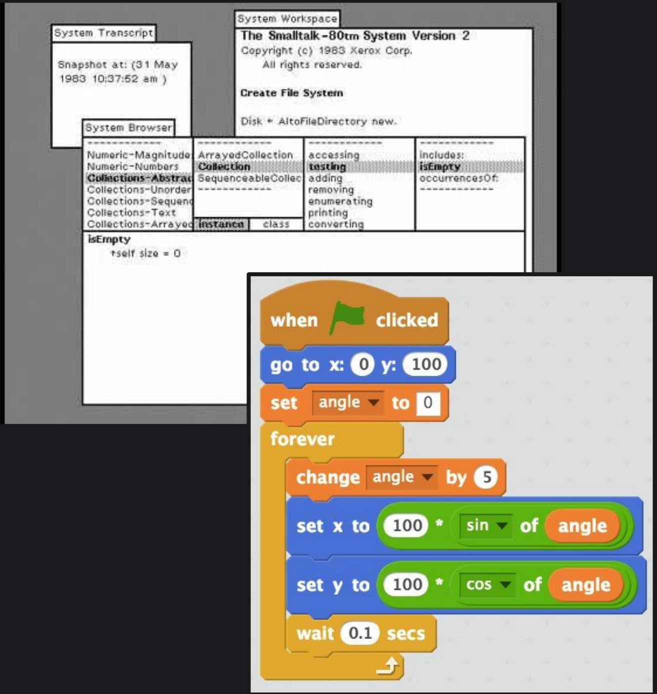
Visual Scripting

Visual scripting is a method to write code that abstracts code into visuals

The idea and implementation have been around for over forty years

Examples include MIT's Scratch and Unreal Engine's Blueprints

You can think of them of really intense IDEs



Snapshot at: (31 May 1983 10:37:52 am)

The Smalltalk-80tm System Version 2
Copyright (c) 1983 Xerox Corp.
All rights reserved.

Create File System

```
Disk ← AltoFileDirectory new.
```

```
isEmpty
    ↑self size = 0
```

```
when 🏁 clicked
go to x: 0 y: 100
set angle ▾ to 0
forever
    change angle ▾ by 5
    set x to (100 * (sin ▾ of angle))
    set y to (100 * (cos ▾ of angle))
    wait 0.1 secs
```

Why Sleek?

Visual languages are usually dismissed as toy languages

They tend to either lack either the breadth or development speed of their textual counterparts

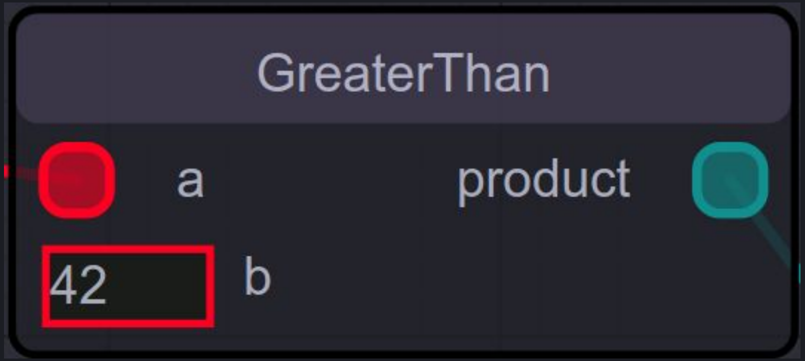
A good union of two of my favorite classes: Interaction Design and Compilers

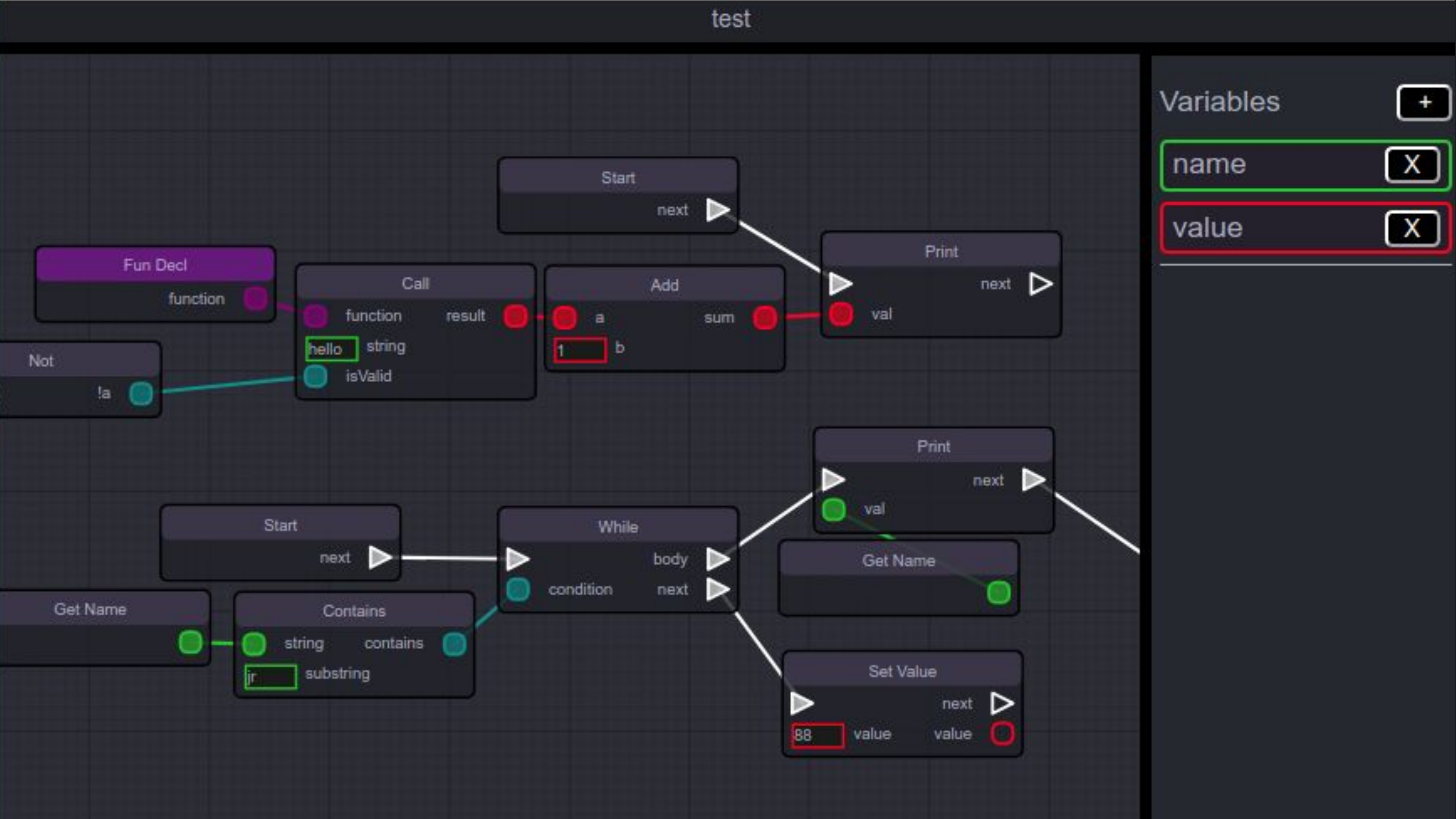
About Sleek

Sleek is an application for developing code in a visual manner

Applications are created by connecting simple expression nodes together to create complex statements

Supports first class functions, multiple returns, object orientation, and functional programming




test
Variables
+
name
X
value
X
Start
next
Print
next
val
Fun Decl
function
Call
function
result
hello
string
isValid
Add
a
sum
1
b
Not
!a
Print
next
val
Start
next
While
body
condition
next
Get Name
Get Name
Contains
string
contains
jr
substring
Set Value
next
88
value
value

Interaction Design

Biggest influences were Unreal's Blueprints and Atom's OneDark UI theme

Making sure there is a balance between learnability and efficiency

Adding in feedback and features for easier development

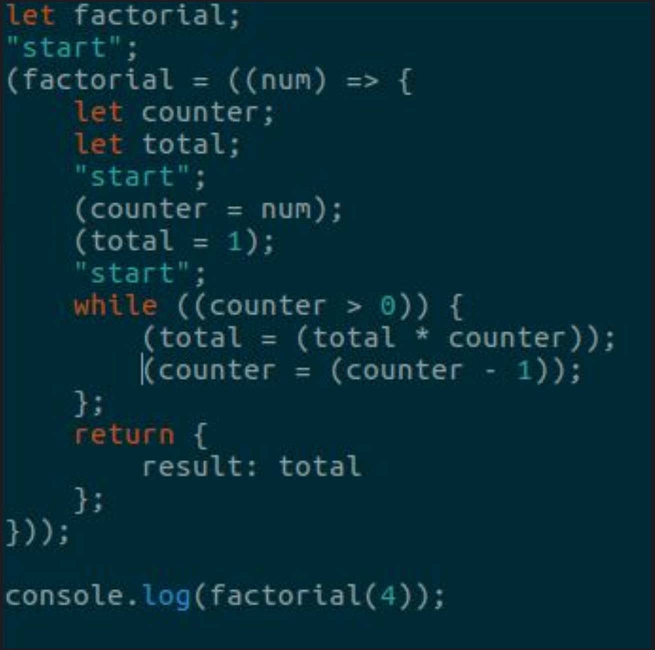
Maximize the amount I utilize the visual nature of the application

Compilers

Sleek is a dynamically strong and static typed language

Makes checks as the code is generated, not at compilation or generation

When code is generated to be run, all Sleek has to do is map nodes to text



```
let factorial;
"start";
(factorial = ((num) => {
    let counter;
    let total;
    "start";
    (counter = num);
    (total = 1);
    "start";
    while ((counter > 0)) {
        (total = (total * counter));
        (counter = (counter - 1));
    };
    return {
        result: total
    };
}));

console.log(factorial(4));
```

Challenges

Visual-JS and weak typing

Restarting application with React

Performing and updating type checks as code is developed

Future Work

Targeting multiple languages for generation

Adding support for multiple files in the project

Simplifying and reworking the live checking system

Questions???