Yo: A video editing language

Mengqing Wang, Munan Cheng, Tiezheng Li, Yufei Ou
Introduction

- Video editing & analytics using yo script:

```python
# This script cuts videos and links them together
a = Clip("helloworld.webm")
b = Clip("helloworld2.webm")
# select part of the video, double represents time in seconds, integer represent frame
e = a[0.0:(1.6 - 0.1)]
f = b[0:1 + 24 * 2]
# concat videos
d = (e & f)
d.save("hello-combined.webm")
```

- Yo is a user-friendly programming language for movie non-linear editing.
Basic

- Function and flow control
  - Level by indentation
- Statically and strongly typed language
  - Type inference
  - Explicit types in function signature

```plaintext
a = 3 + 3.5
# error! Yo does not support type conversion
arr = ["a","b","c"]
for i in arr:
  if i == "a":
    log (i)
  elif i == "b":
    log (i)
    log (i)
# standard output: abb
```

```plaintext
func gcd(a: Int, b: Int) -> Int:
  if (a==0):
    return b
  return gcd(b%a, a)
func compute() -> Int:
  a = 2302
  b = 42
  return gcd(a, b)
compute()
```
Video Editing

- Easy I/O
- Clip operations
  - Access by time or frame index
  - Slicing [ ]
  - Concatenation &
  - Layering ^ @
  - Add key frames . @
- Video analytics
  - Pixel level operation
- Support by libopenshot

```python
1  c = Clip("input.mov")
2  c.alpha@1 = 1.0
3  c.alpha@56 = 0.1
4  c.scale_x@130 = 1.0
5  c.scale_x@210 = 2.0
6  frontend = Clip("overlay.png")
7  frontend.location_x @ 1 = 1.0
8  frontend.location_x @ 150 = 5.8
9  d = c ^ frontend[1:151] @ 3.5
10 e = d[1:50] & d[80:100]
11 e.save("output.webm")
```
Types:

- Decouple video editing functions from core language
- Built-in types (Int, Double, ..., Pixel, Clip)
- User-defined types
- Nested types
- Interaction with existing C++
  - C++ Wrapper
  - Yo header
- Auto memory management

```plaintext
// Type definition
type PixelFilter:
  filtered_count: Int

// Function definitions
func eval(self: PixelFilter) -> PixelFilter:
  self.filtered_count = 0

func filter(self: PixelFilter, p: Pixel) -> Bool:
    self.filtered_count = self.filtered_count + 1
  return true

pf = PixelFilter()
p = Pixel()
pf.filter(p)
log(pf.filtered_count)
```
Architectural Design

- Yo
- Preprocessor
- Scanner
- Semantic Analysis
- TypeReader
- Parser
- Code Generation
- GCC Compilation
Test Plan

- Test suite
  - Unit test:
    - Preprocessor
    - Scanner
    - Parser
    - TypeReader
    - Semantic
    - Code generation
    - Video production
  - Integration test: Log clips
  - Test automation
  - Shell script to automate the flow

```swift
func log_clip(p: Clip) -> Void {
    p.log()
    p.log("helloworld.webm.log")
}
log_clip(Clip("helloworld.webm"))
```
Demo time!

- Demo 1: Flash with Fibonacci
  https://www.youtube.com/watch?v=zFZyuuhah9YI

- Demo 2: Keyframe Animation
  https://www.youtube.com/watch?v=TrA7dJuz9E8

- Demo 3: Time elapsed videos
  https://www.youtube.com/watch?v=rSdKi49fduw