Yo: A video editing language

...

Mengqing Wang, Munan Cheng, Tiezheng Li, Yufei Ou

Introduction

- Video editing & analytics using yo script:

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

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

Basic

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

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

```
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*

```
1  c = Clip("input.mov")
2
3  c.alpha@1 = 1.0
4  c.alpha@56 = 0.1
5
6  c.scale_x@130 = 1.0
7  c.scale_x@210 = 2.0
8
9  frontend = Clip("overlay.png")
10 frontend.location_x @ 1 = 1.0
11 frontend.location_x @ 150 = 5.8
12
13 d = c ^ frontend[1:151] @ 3.5
14 e = d[1:50] & d[80:100]
15
16 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

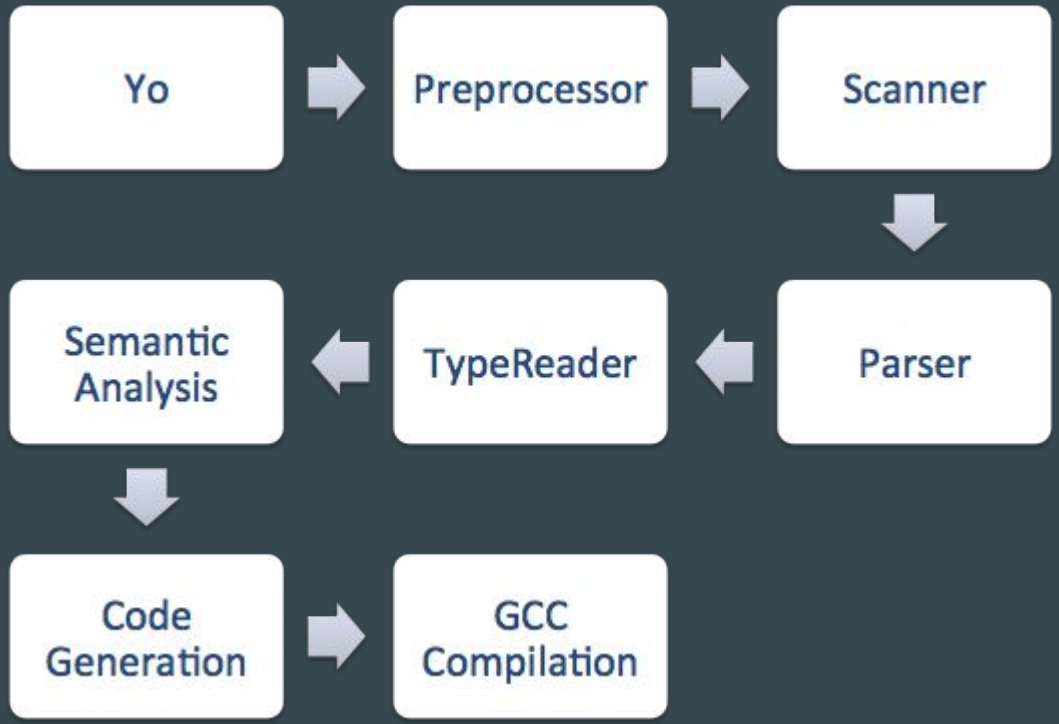
```
type PixelFilter:
    filtered_count: Int

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

    func filter(self: PixelFilter, p: Pixel) -> Bool:
        if p.R == 255 && p.G == 255 && p.B == 255:
            self.filtered_count = self.filtered_count + 1
            return true

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

Architectural Design



Test Plan

```
1  func log_clip(p: Clip) -> Void:
2      p.log()
3      p.log("helloworld.webm.log")
4
5  log_clip(Clip("helloworld.webm"))
```

- 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


```
},
"duration" : 13.29166603088379,
"effects" : [],
"end" : 13.33333301544189,
"gravity" : 4,
"id" : "",
"layer" : 0,
"location_x" : {
   "Points" : [
      {
         "co" : {
            "X" : 1.0,
            "Y" : 0.0
         },
         "interpolation" : 2
      }
   ]
},
"location_y" : {
   "Points" : [
      {
         "co" : {
            "X" : 1.0,
            "Y" : 0.0
         },
         "interpolation" : 2
      }
   ]
},
"perspective_c1_x" : {
   "Points" : [
      {
         "co" : {
            "X" : 1.0,
            "Y" : -1.0
         },
         "interpolation" : 2
      }
   ]
},
```

Demo time!

- Demo 1: Flash with Fibonacci

  https://www.youtube.com/watch?v=zFZyuuah9YI

- Demo 2: Keyframe Animation

  https://www.youtube.com/watch?v=TrA7dJuz9E8

- Demo 3: Time elapsed videos

  https://www.youtube.com/watch?v=rSdKi49fduw