NumNum

A matrix manipulation language

Meet our team

Art Zuks:

- System Architect
- CSMS

Kaustubh Gopal Chiplunkar:

- Language Guru
- MSCE

Paul Czopowik:

- Manager
- GS
- CS Major

Sharon Chen:

- Tester
- SEAS '19
- CS major

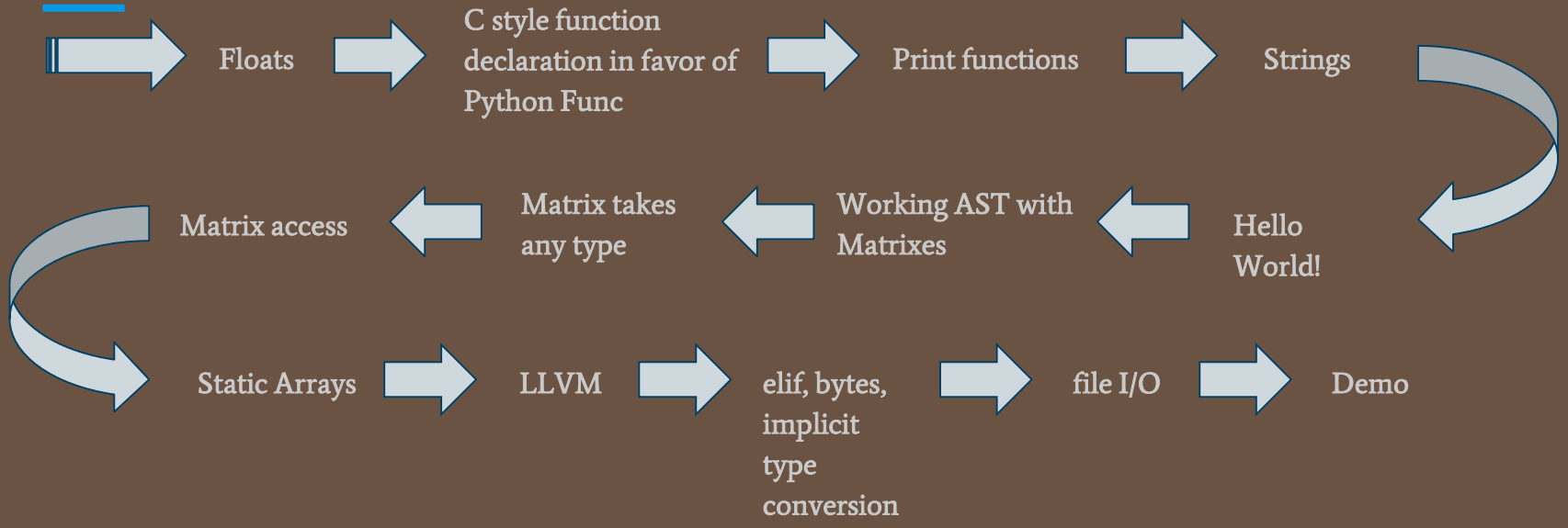
David Tofu:

- Tester
- SEAS '19

Project Planning

We used many cutting-edge tools to help manage our project workflow

Development Timeline



Test Suite

About the Test Suite

How We Wrote It

- While we coded
- For corner cases
- For semantic checking, codegen, pretty-printing
- For both our syntax and language logic
- Wrote fails
- Built upon the Micro-C test suite

An Example

Element-Wise Multiplication
Semantic Checking Tester

```
1
2 int main(){
3     int [10][2] a;
4     int [10][2] b;
5     int [10][2] c;
6     el_mul(a, b, c);
7     return 0;
8 }
```

```
2 int main(){
3     int [10][2] a;
4     int [10][2] b;
5     float [10][2] c;
6     el_mul(a, b, c);
7     return 0;
8 }
```

```
plt4115@plt4115:~/numnum$ ./numnum -a <tests/test-matrix3.num
Fatal error: exception Failure("incompatibles types of matrices to el_mul()")
```

numnum

A simple C-like matrix manipulation language

- Stack allocated arrays support Matrix
- File I/O
- Implicit casting

Quick guide for Programming in numnum

```
string path;

int main() {
  int[3][10][10] a;
  int i;
  path = "./pathtoimage.ppm";
  read(path, a);
}
```

```
int main()
{
 byte a; (* 8 bits *)
 int b; (* 32 bits *)
 float c; (* 64 bits *)
 string d; (* global string *)
 bool e; (* 1 bit *)
}
```

```
int cond(bool b)
{
  int x;
  if (b){
    x = 42;
  }
  elif(true){
    x = 95;
  }
  elif(false){
    x = 423;
  }
  else{
    x = 17;
  }
  return x;
}

int main()
{
 print(cond(true));
 print(cond(false));
 return 0;
}
```

File IO

- Reads into byte, int and float matrix from a binary file
- Writes out matrices to files
- Properly closes file descriptors for the user

Read

```
| A.Call ("read", ([ e ; e2 ])) ->
        let ev = expr builder e and
         ev2 = A.string_of_expr e2 in
        let arrptr = (lookup ev2) in
        let arrtype = (lookup_type ev2) in
        let arrsize = (List.fold_left (fun acc el -> acc*el) 1 (lookup_dims ev2))  in
        let fd = (L.build_call open_func [| ev ; L.const_int i32_t 0|] "open" builder) in
        let ret = (match arrtype with
                  A.Byte -> (L.build_call readbyte_func
                                  [| fd ;
                                    (L.build_gep arrptr [|L.const_int i32_t 0;L.const_int i32_t 0|] "tmp" builder)
                                     L.const_int i32_t (arrsize)|] "read" builder)
                  | A.Int -> (L.build_call read_func
                                  [| fd ;
                                    (L.build_gep arrptr [|L.const_int i32_t 0;L.const_int i32_t 0|] "tmp" builder)
                                     L.const_int i32_t (arrsize*4)|] "read" builder)
                  | A.Float -> (L.build_call readfl_func
                                  [| fd ;
                                    (L.build_gep arrptr [|L.const_int i32_t 0;L.const_int i32_t 0|] "tmp" builder)
                                     L.const_int i32_t (arrsize*8)|] "read" builder)
                  | _ -> raise (Failure ("Unable to read into matrix type " ^ (A.string_of_typ arrtype)))
        ) in
        (ignore (L.build_call close_func [| fd |] "close" builder));ret
```

Write

```
| A.Call ("write", ([e; e2])) ->
        let path = expr builder e and
        var_name =  A.string_of_expr e2 in
        let arrptr = (lookup var_name) in
        let arrsize = (List.fold_left (fun acc el -> acc*el) 1 (lookup_dims var_name)) in
        let fd = (L.build_call creat_func [| path ; L.const_int i32_t 438|] "creat" builder) in
        let ret = L.build_call write_func
                                  [| fd ;
                                    (L.build_gep arrptr [|L.const_int i32_t 0;L.const_int i32_t 0|] "tmp" builder)
                                     L.const_int i32_t (arrsize)|] "write" builder
        in
        (ignore (L.build_call close_func [| fd |] "close" builder));ret
```

```
1  string path;
2
3  int main() {
4    int[3][10][10] a;
5    int i;
6    path = "./pathtoimage.ppm";
7    write(path, a);
8  }
```

Type Casting

- Completely implicit
- Converts to the type that is being assigned to
- For binary operations converts right side to left side type

Operation Casting

```
1  int main() {
2    byte a; (* 8 bit integer *)
3    int b; (* 32 bit integer *)
4    float c; (* 64 bit floating point *)
5    a = b - c; (* operation casting *)
6    return 0;
7  }
```

```
let integer_conv_op lh rh builder =
    let rht =  (L.type_of rh) in
    let lht=  (L.type_of lh) in
    ( match lht with
        | _ when lht == i8_t ->  (
          match  rht with
            | _ when rht == i32_t -> (L.build_intcast rh i8_t "conv" builder)
            | _ when rht == float_t ->  (L.build_uitofp rh i8_t "conv" builder)
            | _ -> rh )
        | _ when lht == i32_t -> (
          match  rht with
            | _ when rht == i8_t -> (L.build_intcast rh i32_t "conv" builder)
            | _ when rht == float_t ->  (L.build_fptosi rh i32_t "conv" builder)
            | _ -> rh )
        | _ when lht == float_t ->  (
          match  rht with
            | _ when rht == float_t -> rh
            | _ ->  ( L.build_sitofp rh float_t "conv" builder) )
        | _ -> rh  ) in
```

Assign Casting

```
let integer_conv_op lh rh builder =
    let rht =  (L.type_of rh) in
    let lht=  (L.type_of lh) in
    ( match lht with
        | _ when lht == i8_t ->  (
          match  rht with
            | _ when rht == i32_t -> (L.build_intcast rh i8_t "conv" builder)
            | _ when rht == float_t ->  (L.build_uitofp rh i8_t "conv" builder)
            | _ -> rh )
        | _ when lht == i32_t -> (
          match  rht with
            | _ when rht == i8_t -> (L.build_intcast rh i32_t "conv" builder)
            | _ when rht == float_t ->  (L.build_fptosi rh i32_t "conv" builder)
            | _ -> rh )
        | _ when lht == float_t ->  (
          match  rht with
            | _ when rht == float_t -> rh
            | _ ->  ( L.build_sitofp rh float_t "conv" builder) )
        | _ -> rh  ) in
```

```
1  int main() {
2    byte a; (* 8 bit integer *)
3    int b; (* 32 bit integer *)
4    float c; (* 64 bit floating point *)
5    a = 1; (*int to byte assignment *)
6    c = a; (*byte to float assignment *)
7    return 0;
8  }
```

Demo 1

Image Manipulation

Image Transformation

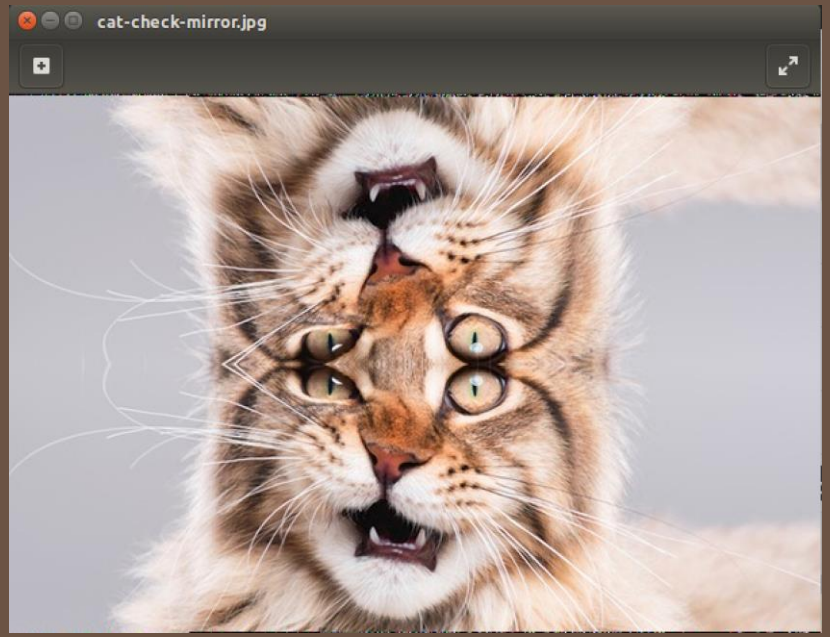
- Colored to BW image conversion
- Blurring with a Gaussian Blur filter
- Edge detection using kernels

Image Transformation

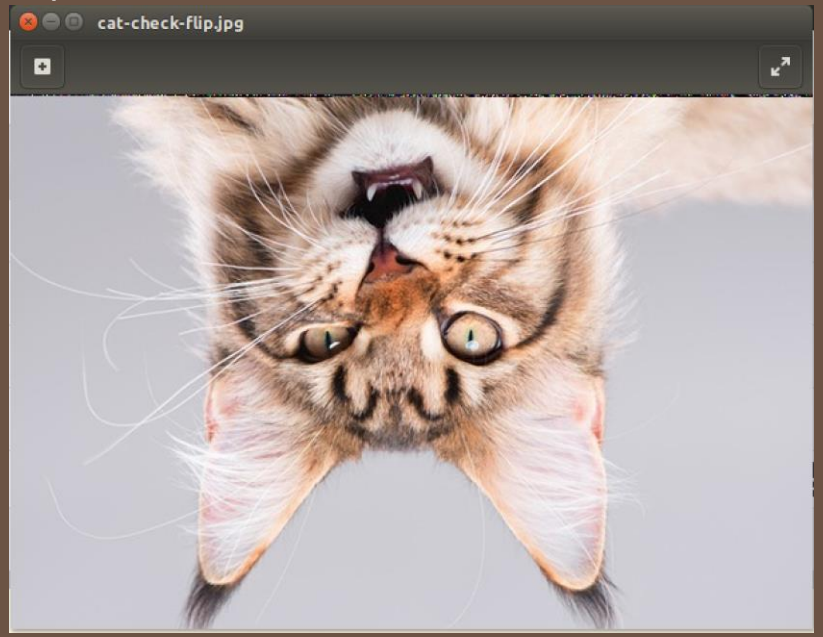
- Using a python script we strip off the headers of the image
- Read the image into an array
- Read the RGB values into float temp variables, using implicit type conversion
- Perform a weighted sum of rgb values
- Assign the float sum back to all the rgb values
- Write back to the same image

Mirror

Reflection



Flip



Demo 2

Optical Character Recognition (OCR) with MNIST

Neural Network Character Recognition

- Used the MNIST database to train a simple neural network for handwritten image recognition
- Used File IO to load the trained weights in the program
- Again used File IO to load the image RGB values (in bytes)
- Performed Accumulation and multiplication of the floats
- Go through all the confidence values, search for the biggest one, display the result