# PANAMX

A matrix manipulation language

James Wong (tw2686)
Project Manager

Brian Hwang (bh2656)
Language Guru

Xingchen Li (xl2801)
System Architect

Zachary Boughner (ztb2003)
Tester

## Motivation: PANAMX

- Common usage of matrices
- Broad range of applications
- Lack of built-in features
- Lightweight,  intuitive language

## Project Workflow: Tools

vmware®

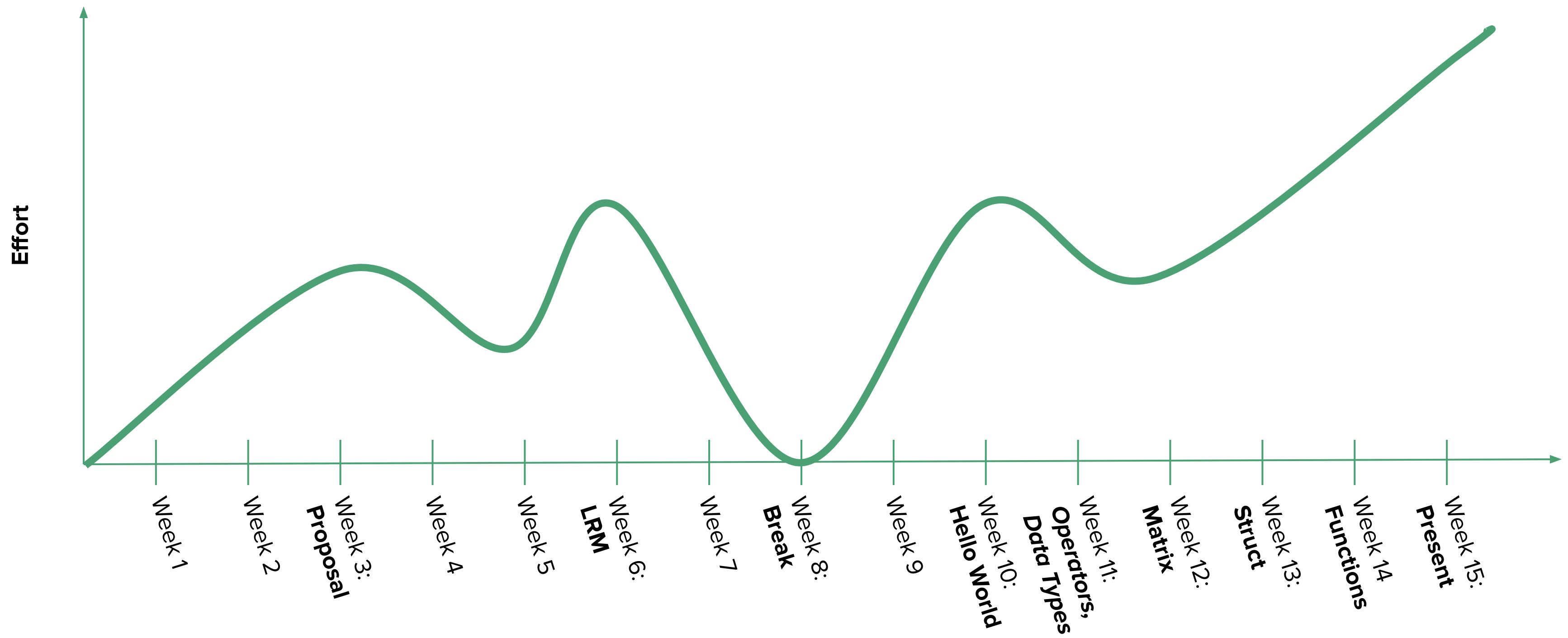
C
Programming

LLVM

OCaml

GitHub

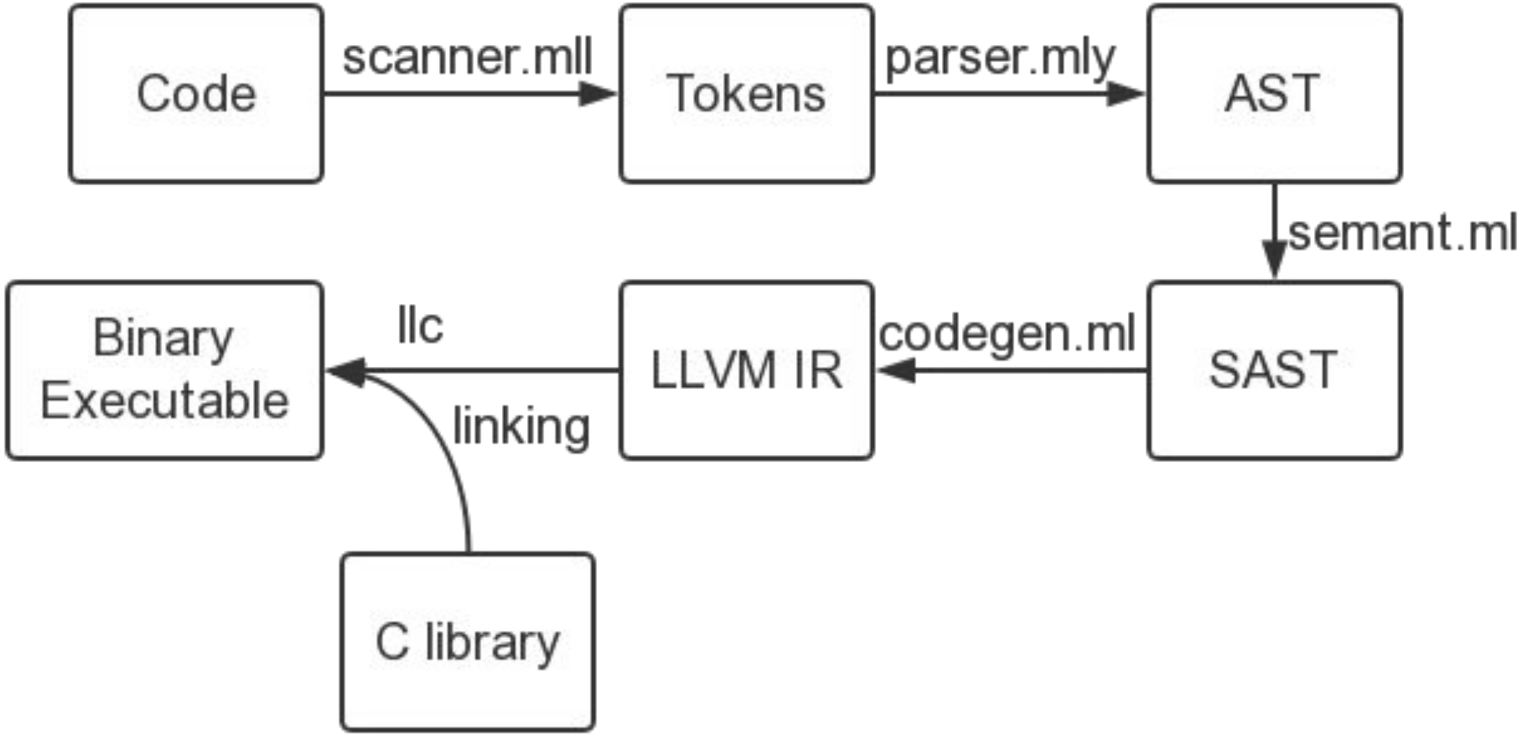
# Project Workflow: Timeline



## Language Overview

- C-like syntax
- Matrix data type
- Struct data type
- Imperative
- Statically-scoped
- Statically-typed

# PANAMX Architecture



## Programming in PANAMX

Primitives:
int, double, bool, void, string, matrix, struct

Control Flow:
if, else, for, while, return

Logical Operators:
!, &&, ||

Arithmetic Operators:
+, -, *, /, =, ++, --

Comments:
// This is a single-line comment
/* This is a multi-line  comment */

Conditional Operators:
==, !=, >, <, >=, <=

Variable Declaration:
```
int a;
```

Variable Initialization:
```
a = 0;
```

Built-In Math Functions:
sqrti( )
sqrtd()
nrooti( )
nrootd()
absi()
absd( )
poweri(r, n)
powerd(r, n)

# Matrix

Description:

- Allows user to build an *n x m* matrix of double types
- Matrix type is defined in C library, and linked to the LLVM

```
typedef struct Matrix {
    int row;
    int col;
    double **mat;
} *matrix;
```

Matrix Declaration:

```
matrix m;
m = <2, 3>; // initialize 2 x 3 matrix of zeros
```

Matrix Initialization:

```
m = [1, 2; 3, 4];
```

Matrix Access:

```
double a;
a = m[0][1];
```

Matrix Slicing:

```
printm(m[1:2][1:2]); // print “[4]”
```

# Built-in matrix functions:

| | |
|---|---|
| matrixHeight(matrix m) | returns the number of rows in the matrix |
| matrixWidth(matrix m) | returns the number of columns in the matrix |
| sum(matrix m) | returns the sum of all elements in the matrix |
| mean(matrix m) | returns the mean of all elements in the matrix |
| trans(matrix m) | returns the transposed version of the matrix |
| det(matrix m) | computes the determinant of the matrix |
| rank(matrix m) | returns the rank of the matrix |
| rref(matrix m) | returns the matrix in reduced row echelon form |
| .* | element-wise multiplication of matrices |
| ./ | element-wise division of matrices |

Matrix Operators:
+, -, *

```
matrix m;
matrix n;
m = [1, 2;
     3, 4;
     5, 6];
n = [1, 2, 3;
     4, 5, 6];
m = m * n;
printm(m);
[
      9.000   12.000   15.000
      19.000  26.000   33.000
      29.000  40.000   51.000
]
```

## User Defined Types: Struct

Our language support struct that allows user to define customized data structure

```
struct Node {
    int aa;
    bool bb;
    double cc;
    matrix dd;
};
```

This defines a structure with an int type, a bool type, a double type, and a matrix type

struct declaration & initialization

```
struct Node n;
n = <struct Node>;
```

how to access the member in the struct?

```
print(n.aa);
n.cc = 1.23;
n.dd = [1, 2;
        3, 4];
```

# Semantic Checks

```
int main(){
   matrix m;
// m consists of str, double
   m = [“s”, 3.3;
        2.3, 3.4];
      return 0;
}
```

```
###### Testing test-matrixerror
./panamx.native tests2/px_tests/test-matrixerror.px > test-matrixerror.ll
Fatal error: exception Failure("matrix elements can only be int/double type")
###### FAILED
```

```
int main(){
   matrix m;
// Define 3 x 3 matrix
   m = [1, 3.3, 2.2;
        2, 3.4, 1.2;
        4, 1.4, 4.4];
// [3][3] is out of boundary
   printf(m[3][3]);
   return 0;
}
```

```
###### Testing test-matrixerror
./panamx.native tests2/px_tests/test-matrixerror.px > test-matrixerror.ll
llc -relocation-model=pic test-matrixerror.ll > test-matrixerror.s
cc -o test-matrixerror.exe test-matrixerror.s matrix.o
./test-matrixerror.exe
matrix index out of bound
###### FAILED
```

## Demo

## Lessons Learned

- Project management between **different development environments** can be ***hard***; we probably should have used Docker
- It makes life easier to do **robust testing** consistently throughout development rather than at the end of a long commit
- Always **ask questions**: even if it's a "stupid" question, having a colleague quickly answer that question saves the group more time than having you struggle to figure it out yourself
- There are *always* more **test cases**... Think creatively to cover all your bases
- It's sometimes better to **start coding** instead of talking about coding
- Even with a good version control system, **communication** is key for not repeating work

## Thank You!

Special Thanks To Our TA Ryan Bernstein!