MAP

shortestPathTester.map

```
func main(){
Graph g = new Graph();
Node la = new Node({'temp':90,'humidity':'low'});
Node sj = new Node({'temp':50,'humidity':'low'});
Node sf = new Node({'temp':65, 'humidity':'high'});
UnDirEdge la_sj = new UnDirEdge(la, sj, {'cost': 100});
UnDirEdge sj_sf = new UnDirEdge(sj, sf, {'cost':50});
UnDirEdge la_sf = new UnDirEdge(la, sf, {'cost':180});
g.findShortest(la, sf, 'cost');
}
```






How MAP is different...

- has edges, nodes, graphs, and paths as inherent data structures
- is easier and more accessible for beginner programmers
- abstracts away syntax and set-up without loss of functionality

Why use MAP instead?

- Graph programming is difficult and complicated in other languages
- Other languages don't have basic graph data structures such as edges and nodes as inherent data types, forcing the programmer to build them themselves


MAP: The Graph Programming Language

Serena Simkus - Project Manager
Sandya Sankarram - Language Guru
Alex Merkulov - System Architect
Alfred Tan - System Integrator
Tommy Inouye - Verification & Validation Person

Language Syntactic Structure

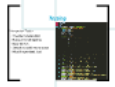
- Basic data types: Text, Numeric, Boolean
- Derived Data Type examples:
  - Graph g = new Graph();
  - Node la = new Node({'temp':90});
  - DirEdge la_sj = new DirEdge(la, sj, {'cost': 100});
- Scoped with curly braces
- If, for, foreach
- Reserved words and characters

MAP is...

Simple
Familiar
Reusable
Imperative
Intuitive








Project Management Tools

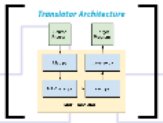
asana
Gmail


Asana, Google groups, Gmail, texting, and GitHub

MAP

shortestPathTester.map

```
func main(){
Graph g = new Graph();
Node la = new Node({'temp':90,'humidity':'low'});
Node sj = new Node({'temp':50,'humidity':'low'});
Node sf = new Node({'temp':65, 'humidity':'high'});
UnDirEdge la_sj = new UnDirEdge(la, sj, {'cost': 100});
UnDirEdge sj_sf = new UnDirEdge(sj, sf, {'cost':50});
UnDirEdge la_sf = new UnDirEdge(la, sf, {'cost':180});
g.findShortest(la, sf, 'cost');
}
```

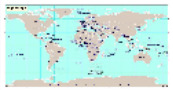


How MAP is different...

- has edges, nodes, graphs, and paths as inherent data structures
- is easier and more accessible for beginner programmers
- abstracts away syntax and set-up without loss of functionality

Why use MAP instead?

- Graph programming is difficult and complicated in other languages
- Other languages don't have basic graph data structures such as edges and nodes as inherent data types, forcing the programmer to build them themselves



MAP: The Graph Programming Language

Serena Simkus - Project Manager
Sandya Sankarram - Language Guru
Alex Merkulov - System Architect
Alfred Tan - System Integrator
Tommy Inouye - Verification & Validation Person

Language Syntactic Structure

- Basic data types: Text, Numeric, Boolean
- Derived Data Type examples:
  - Graph g = new Graph();
  - Node la = new Node({'temp':90});
  - DirEdge la_sj = new DirEdge(la, sj, {'cost': 100});
- Scoped with curly braces
- if, for, foreach
- Reserved words and characters

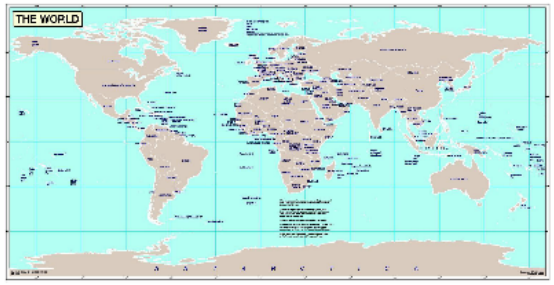
MAP is...

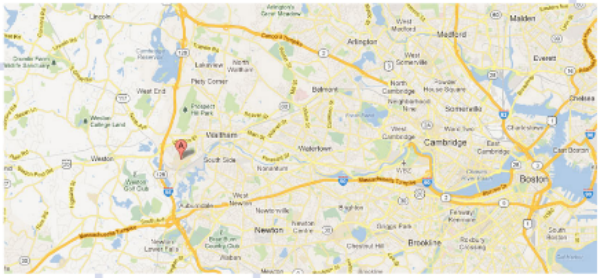
Simple
Familiar
Reusable
Imperative
Intuitive

Project Management Tools

asana
Gmail

Asana, Google groups, Gmail, texting, and GitHub





MAP: The Graph Programming Language

Serena Simkus - Project Manager
Sandya Sankarram - Language Guru
Alex Merkulov - System Architect
Alfred Tan - System Integrator
Tommy Inouye - Verification & Validation Person

MAP is...

Simple

Familiar

Reusable

Imperative

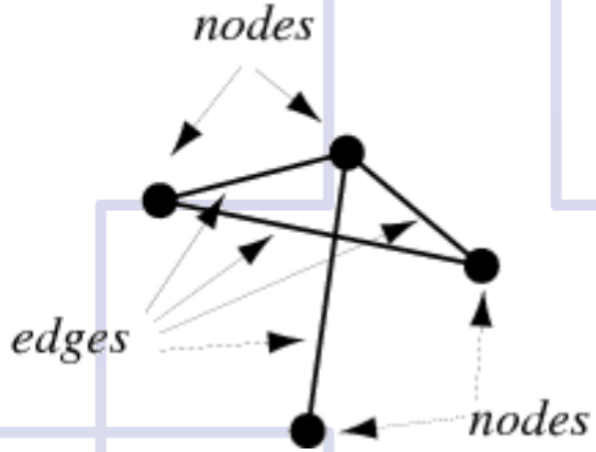
Intuitive

Why use MAP instead?

- **Graph programming is difficult and complicated in other languages**
- **Other languages don't have basic graph data structures such as edges and nodes as inherent data types, forcing the programmer to build them themselves**

How MAP is different...

- **has edges, nodes, graphs, and paths as inherent data structures**
- **is easier and more accessible for beginner programmers**
- **abstracts away syntax and set-up without loss of functionality**



Language Syntactic Structure

- Basic data types: Text, Numeric, Boolean
- Derived Data Type examples:
  - Graph g = new Graph();
  - Node la = new Node({'temp':90});
  - DirEdge la_sj = new DirEdge(la, sj, {'cost': 100});
- Scoped with curly braces
- if, for, foreach
- Reserved words and characters

Built-in Functions

Graph.add(node)
Graph.delete(node)
Graph.addEdge(edge)
Graph.findShortest(node1, node2, attribute)
Graph.draw(filename)
... among others!

shortestPathTester.map

```
func main(){
Graph g = new Graph();
Node la = new Node({'temp':90,'humidity':'low'});
Node sj = new Node({'temp':50,'humidity':'low'});
Node sf = new Node({'temp':65, 'humidity':'high'});
UnDirEdge la_sj = new UnDirEdge(la, sj, {'cost': 100});
UnDirEdge sj_sf = new UnDirEdge(sj, sf, {'cost':50});
UnDirEdge la_sf = new UnDirEdge(la, sf, {'cost':180});
g.findShortest(la, sf, 'cost');
}
```

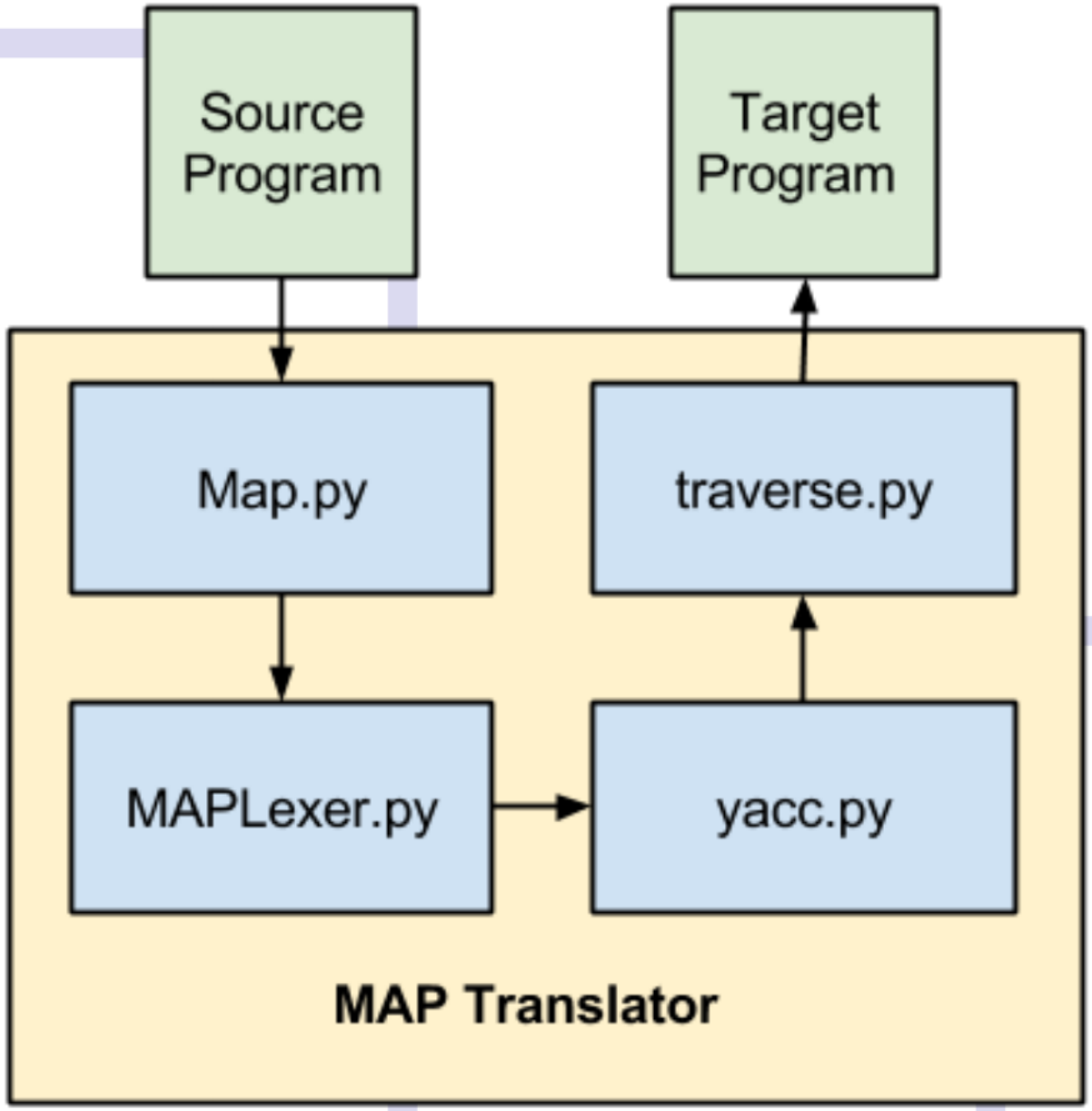
Project Management Tools

asana:

Gmail by Google

Asana, Google groups, Gmail, texting, and GitHub

Translator Architecture



Translator Architecture

- Map.py: indentation, scope
- MAPLexer.py: lex rules
- yacc.py: grammar, AST
- Traverse.py: AST->python

Run-time Environment

pip install virtualenv
virtual env
source env/bin/activate
pip install ply
pip install asciitree
pip install networkx
pip install -U pyyaml nltk

```
dyn-160-39-247-224:~ alfredtan$ cd map
dyn-160-39-247-224:map alfredtan$ python Map.py test/sample2.map
```

Software Development Environment

- Python 2.7.5
- GCC 4.2.1
- Mac OS X

Compiler-Generator Tools

- **PLY: Python Lex-Yacc**

Testing

Lexer Tester

- Ensured the effectiveness of lex
- Mostly ensured the flexibility of ID
- Ran a test to check tokens
- Ensures consistent tokens
- Easy to add new tokens to test

```
dyn-160-39-231-16:map tommyinouye$ python lexertest.py
Asserting syntax of hello world and checking generated tokens:
Hello world test passed!

.Testing all tokens:
asserting key: 1234
asserting key: elif
asserting key: findShortest
asserting key: false
asserting key: read
asserting key: Graph
asserting key: else
asserting key: in
asserting key: .01231
asserting key: equals
asserting key: break
asserting key: func
asserting key: Time
asserting key: path
asserting key: 12312.213123
asserting key: true
asserting key: addEdge
asserting key: if
asserting key: Node
asserting key: write
asserting key: null
asserting key: piNULLdel
asserting key: for
asserting key: deleteEdge
asserting key: Text
asserting key: getEdge
asserting key: Numeric
asserting key: 1
asserting key: return
asserting key: add
asserting key: continue
asserting key: foreach
asserting key: adjacent
asserting key: Path
asserting key: include
asserting key: delete
Token test passed!
.
----------------------------------------------------------------------
Ran 2 tests in 0.010s

OK
```

Testing

Traverse Tester

- Tested compiler
- Looks for syntax errors
- Ensured consistency
- Easy to run
- Simple to add more tests

```
dyn-160-39-231-16:map tommyinouye$ python traverseTester.py
BEGIN TRAVERSE TEST
-------------------
Testing for each statements:
Test for each statement: Passed
.Testing for statements:
Test for statement: Passed
.Testing helloworld:
Test hello world: Passed
.testing if elif else statements:
Test if elif else test: Passed
.Testing if and else statements:
Test if else test: Passed
.Testing if statements:
Test if statement: Passed
..Test nodes:
Test simple node: Passed
.testing paths:
Test building path: Path
.Testing print statements:
Test print statement: Passed
.Test read and write:
Test read and writing: Passed
.testing sample program #1
Test sample program 1: Passed
.Testing sample program #2
Test sample program 2: Passed
.Testing sample program #3
Test sample program 3: Passed
.Testing sample program #4
Test sample program 4: Passed
.Testing sample program #5
Test sample program 5: Passed
.Testing sample program #6
Test sample program 6: Passed
.Testing shortest path
Test shortest path test: Passed
.
----------------------------------------------------------------------
Ran 18 tests in 0.072s

OK
```

Testing

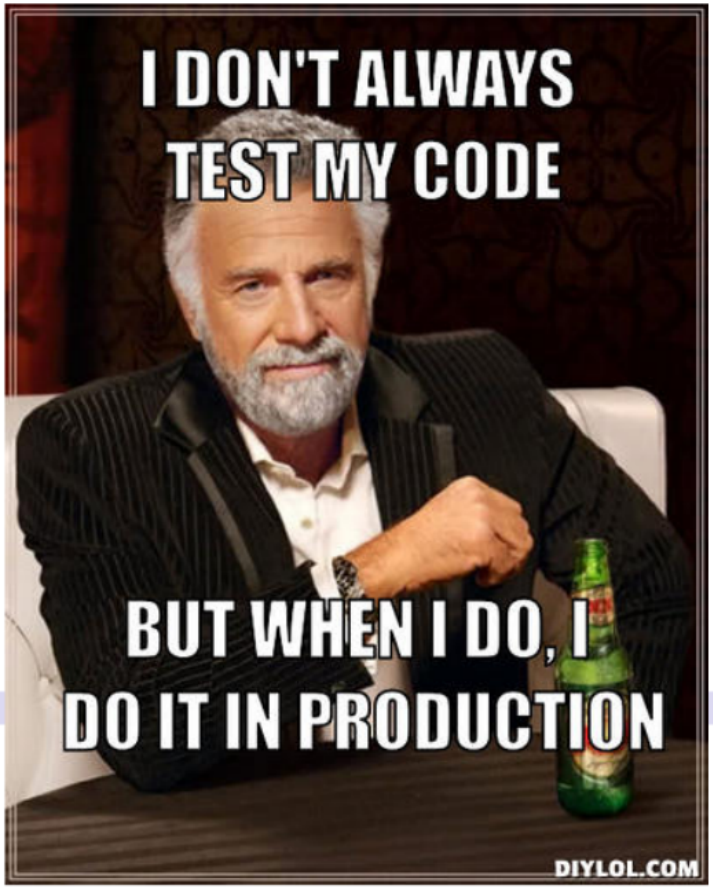
Interpreter Tester

- Checks indentation
- Runs error checking
- Easy to run
- Simple to add more tests
- Most important test

```
import unittest
import sys
import os
class TestInterpreterSyntax(unittest.TestCase):
    def assert_prog(self, testname, outname):
        t=open(testname,'r')
        o=open(outname,'r')
        teststr=t.read()
        outstr=o.read()
        self.assertEqual(teststr,outstr)
    def testhelloworld(self):
        os.system("python Map.py test/helloworld.map")
        self.assert_prog("test/helloworld.py","test/helloworld.out")
    def testfactorial(self):
        os.system("python Map.py test/factorial.map")
        self.assert_prog("test/factorial.py","test/factorial.out")
    def testforeach(self):
        os.system("python Map.py test/foreachtest.map")
        self.assert_prog("test/foreachtest.py","test/foreachtest.out")
    def testforstatement(self):
        os.system("python Map.py test/forstatementtest.map")
        self.assert_prog("test/forstatementtest.py","test/forstatementtest.out")
    def testifelifelse(self):
        os.system("python Map.py test/ifelifelsetest.map")
        self.assert_prog("test/ifelifelsetest.py","test/ifelifelsetest.out")
    def testifelse(self):
        os.system("python Map.py test/ifelsetest.map")
        self.assert_prog("test/ifelsetest.py","test/ifelsetest.out")
    def testif(self):
        os.system("python Map.py test/iftest.map")
        self.assert_prog("test/iftest.py","test/iftest.out")
    def testnode(self):
        os.system("python Map.py test/nodetest.map")
        self.assert_prog("test/nodetest.py","test/nodetest.out")
    def testsample1(self):
        os.system("python Map.py test/sample1.map")
        self.assert_prog("test/sample1.out","test/sample1.py")
    def testincludeinput(self):
        os.system("python Map.py test/testincludeinput.map")
        self.assert_prog("test/testincludeinput.out","test/testincludeinput.py")
    def testpathtest(self):
        os.system("python Map.py test/pathtest.map")
        self.assert_prog("test/pathtest.py","test/pathtest.out")
```

Demo!

Here is a cool program that displays a graph of our team!



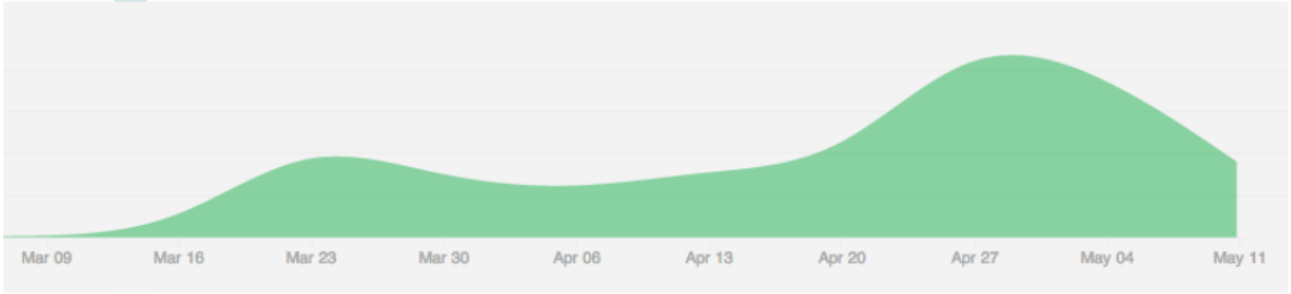
Conclusions

What we learned:

- Progress is important!! Keep moving forward
- GitHub, version control

What went well:

- Meetings!
- Communication
- Dividing up work



What we would have done differently:

- Start implementation earlier
- Working kernel earlier

Why MAP?

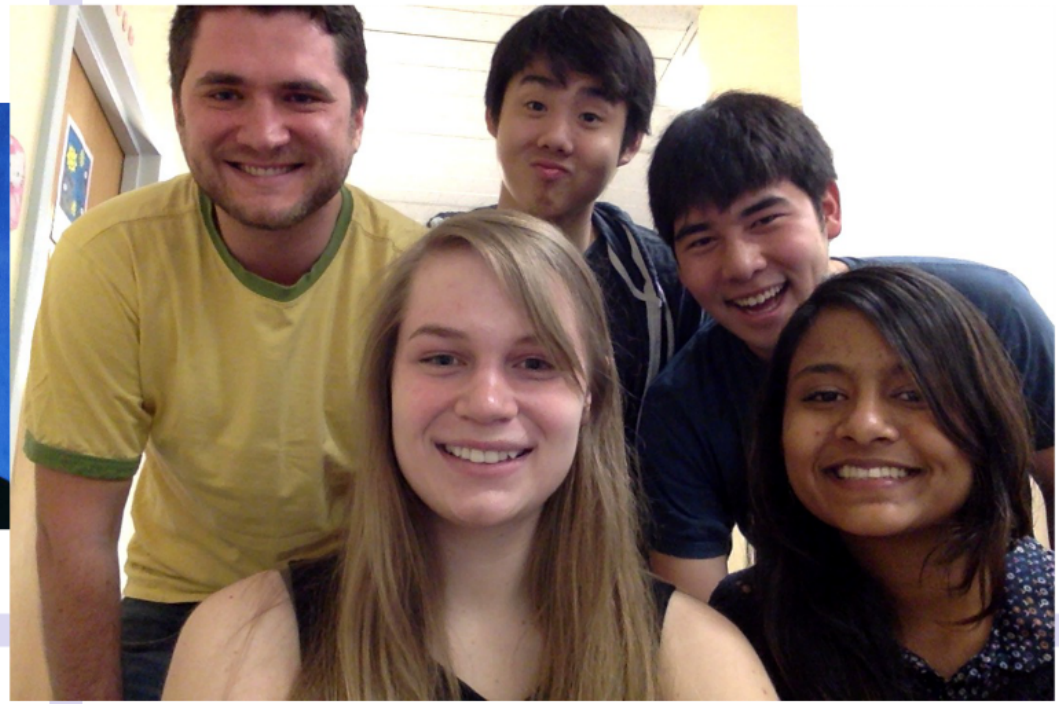
- Did you see that awesome demo?!?
- Graph programming made easier.

Thank you for listening!! :)

Repository:

https://github.com/PLT-MAP/map

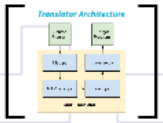
The Team:



MAP

shortestPathTester.map

```
func main(){
Graph g = new Graph();
Node la = new Node({'temp':90,'humidity':'low'});
Node sj = new Node({'temp':50,'humidity':'low'});
Node sf = new Node({'temp':65, 'humidity':'high'});
UnDirEdge la_sj = new UnDirEdge(la, sj, {'cost': 100});
UnDirEdge sj_sf = new UnDirEdge(sj, sf, {'cost':50});
UnDirEdge la_sf = new UnDirEdge(la, sf, {'cost':180});
g.findShortest(la, sf, 'cost');
}
```

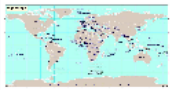


How MAP is different...

- has edges, nodes, graphs, and paths as inherent data structures
- is easier and more accessible for beginner programmers
- abstracts away syntax and set-up without loss of functionality

Why use MAP instead?

- Graph programming is difficult and complicated in other languages
- Other languages don't have basic graph data structures such as edges and nodes as inherent data types, forcing the programmer to build them themselves



MAP: The Graph Programming Language

Serena Simkus - Project Manager
Sandya Sankarram - Language Guru
Alex Merkulov - System Architect
Alfred Tan - System Integrator
Tommy Inouye - Verification & Validation Person

Language Syntactic Structure

- Basic data types: Text, Numeric, Boolean
- Derived Data Type examples:
  - Graph g = new Graph();
  - Node la = new Node({'temp':90});
  - DirEdge la_sj = new DirEdge(la, sj, {'cost': 100});
- Scoped with curly braces
- if, for, foreach
- Reserved words and characters

MAP is...

Simple
Familiar
Reusable
Imperative
Intuitive

Project Management Tools

asana
Gmail

Asana, Google groups, Gmail, texting, and GitHub