Interactive Fiction Language
Team Introduction

Project Manager: John Liu
Language Guru: Matthew Suozzo
System Architect: Michael Yan
System Integrator: Qian Yu
System Tester: Heather Fisher
Interactive Fiction = ?

- Text Adventure
- Story-Driven
- Interactive
Goals of our IF Language

- Easily Create Interactive Fiction
- Minimal Programming Experience Needed
Our User

- Familiar with programming concepts
- Mostly a Game Designer
  - (instead of coder)
Goals of our IF Language

- Easily Create Interactive Fiction
- Minimal Programming Experience Needed
- Decouple Roles of Writer & Programmer
  - eg. Dialogue Trees
Language Design

Matthew Suozzo
Buzzwords

- Keywords
- Concise
- Symbol-less
- Object-oriented
- Plain-English
- Readable
- Python
- Intuitive
What we Got

4 Object Types (TLTs)

<table>
<thead>
<tr>
<th>Item</th>
<th>Trait</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>Setting</td>
</tr>
</tbody>
</table>
What we Got

4 Primitive Types

String          TF (bool)

Integer         Decimal
Language Structure

● Program is list of defined types (TLTs)

ITEM Apple:
{block}:
{statements}

STARTS : Constructor
ACTIONS : Actions available in game
FUNCTIONS : Functions available in the code
DIALOGUE : Defines Dialogue

● PLAYER Character is like main method
Syntax / Statements

ADD item TO char
REMOVE item FROM char
SET val TO 3
PRINT str . "str"
MOVE char TO loc
EXECUTE func WITH arg1
INITIATE DIALOGUE AT #LABEL#
INCREASE val BY 3
DECREASE val BY 1
NUMBER OF item IN char
GOTO #LABEL#
USING "diag.txt"
EXIT

IF case1:
{statements}
ELSE IF case2:
{statements}
ELSE:
{statements}
ADD {STRING s="hello"}
CHARACTER maid:
  “I am a maid of the house.”
START:
  USES “Text.txt”
  ADD apple TO SELF
ACTIONS:
  “talk”
    IF SELF HAS apple:
      INITIATE DIALOGUE AT #LABEL A#
    ELSE:
      INITIATE DIALOGUE AT #LABEL D#
DIALOGUE:
  #LABEL A#:
    IF LAST_INPUT EQUALS “1”:
      GOTO #LABEL B#
    ELSE IF LAST_INPUT EQUALS “2”:
      GOTO #LABEL C#
  #LABEL B#:
    IF LAST_INPUT EQUALS “1”:
      REMOVE apple
      ADD apple TO PLAYER
      GOTO #LABEL C
    ELSE IF LAST_INPUT EQUALS “2”:
      EXIT
Maid: Hi, can I help you?
(1) Ask where you are
(2) Ask for an item

Maid: You are in the bedroom of an old house.
(1) Ask for an item
(2) Exit

The maid gives you an apple
(1) Ask where you are
(2) Exit

Maid: Hi, can I help you?
(1) Ask where you are

Maid: You are in the bedroom of an old house.
(1) Exit
DEMO!

Michael Yan
System Architecture

Michael Yan
CHARACTER John:
START:
  MOVE SELF TO IFL
ACTIONS:
  "talk"
  PRINT "Hello"
END_BLOCK

[  
  CHARACTER, John:
  START:
  MOVE SELF TO IFL
  ACTIONS:
    "talk"
    PRINT "Hello"
  END_BLOCK
]
[  
  CHARACTER,  John:
  START:
  MOVE SELF TO IFL
  END_BLOCK
  ACTIONS:
  "talk"
  PRINT
  "Hello"
  END_BLOCK
]
def p_move(p):
    'move : MOVE object_chain TO object_chain'
    p[0] = (p[1], p[2], p[4])

('John', None,
 ('START',
  ('MOVE',
   ('OBJ', 'SELF'),
   ('OBJ', 'IFL')
  ),
  ('ACTIONS',
   ('talk',
    ('PRINT',
     ('Hello'))
  ),
  None,
  None
 )
)
Program

TLT (John)

start

Statement

(MOVE)

('OBJ',  'SELF'),

('OBJ', 'IFL')

actions

Action (talk)

Statement

PRINT ('Hello')
class John:
    def __init__(self):
        self.location = 'IFL'
    def talk():
        print 'Hello'

character John:
    start:
        move self to IFL
    actions:
        "talk"
        print "Hello"
System Integration

Qian Yu
How The Pieces Fit

preprocessor.py
ifl_lex.py ifl_yacc.py
generator.py

compiler.py

./ifl

your_file.ifl

game/*
What's Actually Generated?

- a game/ directory
- a .py class file for each object type (Item, Character, Trait, Setting)
- a main game file (game.py) that actually runs the game
Execution Environment

- game.py
- instantiate and setup characters and settings
- Enter into read-evaluate-print-loop (REPL)
  - Get User Command
  - Search for Command in Action List of Current Setting
  - Call Appropriate Functions
  - Print Results
  - Update Available Actions
  - Repeat
Development Environment

- Python Lex/Yacc (PLY)
- Git and Github for Version Control
- IDE/Debugger: PyCharm
- Terminal for Running and Testing
Testing

Heather Fisher
Testing Process

- Unit Test
- Test as we go
Test Suite

class SuccessTest(unittest.TestCase):
    def test_suc1(self):
        os.system("python compiler.py examples/ex1.ifl")
    def test_suc2(self):
        os.system("python compiler.py examples/ex2.ifl")
    def test_suc3(self):
        os.system("python compiler.py examples/ex3.ifl")

class ExpectedFailureTestCase(unittest.TestCase):
    def test_fail(self):
        os.system("python compiler.py test/test4.ifl")
    def test_fail2(self):
        os.system("python compiler.py test/test5.ifl")
    def test_fail3(self):
        os.system("python compiler.py test/test6.ifl")
    def test_fail4(self):
        os.system("python compiler.py test/test7.ifl")
    def test_file5(self):
        os.system("python compiler.py test/test8.ifl")
    def test_file6(self):
        os.system("python compiler.py test/test9.ifl")
Sample Test Program

Will this program work?
Results

- Missing
  
  \texttt{ADD\{INTEGER max = 100\} TO SELF}
Why Testing is Important

- Test early and test often!
- You never know what will break your code
Project Management

John Liu
Project Process

- Waterfall Methodology to Agile Development
- Python & Java to Python Only
- Team Organization
  - Weekly Meetings, Google Hangout, Instant Messaging
  - Google Docs, Github
Lessons Learned

● Taking Advantage of Version Control (Git)
● Group Development vs Working Individually
● Communicate, Communicate, Communicate!
Possible Expansion

- Expand Character Encoding
- Support for Libraries
- Dialog Tree markup
- More Customization Options
  - Support for Multiplayer / Networking
The End

IFL: Do you have any questions?
(1) Yes
(2) No
>> _