Image Processing Language?

Jianning Yue
Wookyyn Kho
Young Jin Yoon
Contents

- IPL?
- Advantages
- Syntax
- Development
- Examples
- Lessons Learned

AGL : Animation applet Generation Language
IPL?

- IPL is not Image Processing Language
  - Now this is an **Animation applet Generation Language**!
Advantages (1/2)

- IPL provide very flexible image handling
  - Provide fundamental operation for image as expression
    - Rotate (@ operator)
    - Translate (' operator)
    - Scale (^ operator)
  - Provide animate() function to produce an animated Image
  - Provide coord type to handle coordinates
Advantages (2/2)

- Easy to learn
  - C like syntax and scope
  - Easily-recognized operator
    - (^ is power operator from another language)
- Productive
  - Can be exported as an JAVA applet
    - smaller than GIF Animated Image
Syntax : Types (1/3)

- Four types in IPL
  - number
  - image
  - coord
  - bool

- Optional declarator in IPL
  - [] for array definitions
Syntax : Types (2/3)

- **FLEXIBLE ARRAY HANDLING (1/2)**

  - For both
    - `imgA[0]`
      ![Diagram for `imgA[0]`]
    - `imgA[0~1]`
      ![Diagram for `imgA[0~1]`]
  
  - For lvalue
    - `imgA[1+]`
      ![Diagram for `imgA[1+]`]
    - `imgA[1~2+]`
      ![Diagram for `imgA[1~2+]`]
Syntax : Types (3/3)

• FLEXIBLE ARRAY HANDLING (2/2)
  • For rvalue
    • imgA[0-]
      [Diagram: Three blue and one white square]
    • imgA[1~2-]
      [Diagram: Two blue and two white squares]
Syntax : Expr (1/4)

• Basic image operator
  • imgA = imgA @ numA; // rotate operator
  • imgA = imgA ^ numA; // scale operator
  • imgA = imgA ` coordA; // set operator
  • imgA = imgA : numA; // alpha operator
  • imgA = imgA $ imgB; // concat operator
Syntax : Expr (2/4)

- Basic bool operator
  - `boolA = numA > numB;`  // gt operator
  - `boolA = numA < numB;`  // lt operator
  - `boolA = numA >= numB;`  // ge operator
  - `boolA = numA <= numB;`  // le operator
  - `boolA = numA == numB;`  // eq operator
  - `boolA = numA != numB;`  // neq
  - `boolA = !boolA`  // not operator
Basic arithmetic operator

- \( \text{numA} = \text{numA} \times \text{numB}; \) // multiply
- \( \text{numA} = \text{numA} \div \text{numB}; \) // division
- \( \text{numA} = \text{numA} \mod \text{numB}; \) // modulo
- \( \text{numA} = \text{numA} + \text{numB}; \) // plus
- \( \text{numA} = \text{numA} - \text{numB}; \) // minus

For coord, there is no operation. However we can still handle this. How?
For coordination

- cooA = (xof(cooA),numA);
- cooB = (numA, yof(cooA));

By providing xof() and yof(), we can still maintain flexibility without any complexibility!
Syntax : Stmt

- Providing while, if statement just as almost same as C’s statement definition.
  - Except using {} for single statement.

- You can define a function using defunc keywords.
  - defunc foo (number A, number B) number C
    { C = A + B; }

- Providing return, break, continue statements.
Development

- Task Distribution
- Architecture Overview
- Implementation
- Test and Debug plan
Task distribution

Front-end

Parser
Lexer
Walker

Back-end

Animation Module

Test & Integration

Parser
Lexer
Walker

Animation Module

Test & Integration

Young Jin Yoon
Wookyun Kho
Jianning Yue

AGL: Animation applet Generation Language
Architecture Overview

- Used UML Class Diagram
Implementation (1/3)

- ANTLR
  - Parser
  - Lexer
  - Walker
- Animation Module
  - Animation Displaying Engine
  - Animation Applet Code
Implementation (2/3)

AGL: Animation applet Generation Language

Lexer → Token Stream → Parser → AST → AST Walker

Input Stream

Exception Handling

Symbol Table

IPL Object

Non Image Output

Create/Invoke/Fetch/Set

Java 2D Animation module → Display Images

Java 2D Applet module

Save Animation Information

Display Images

Save Animation Information
Implementation (3/3)

- Animation Applet
  - If you do “export”, you have to specify the filename.
  - Ex) export to “IPLoutput.ipl”

```xml
<applet code=IPLApplet.class width=1024 height=600>
  <param name="fps" value="20">
  <param name="ipl" value="IPLoutput.ipl">
</applet>
```
Test and Debug Plan

- Test plan
  - Control statement
  - Function call
  - Static scope
  - Static image display
  - Image rotation
  - Image scale
  - Image rotate
  - Image set
  - Image alpha
  - Image animation
  - Image Array animation
  - Combined Image animation

- Debug Plan
  - Make debug flag and debug() for debugging
  - Using assert()
  - Using eclipse IDE
    - Good for debugging
Examples (1/4)

Basic Arithmetic, Coordination

```plaintext
defunc add (number a, number b) number c { c = a + b; }

number numA, numB;
number[] numC = { 0, 1, 2, 3, 4 }, numD;
coord cooA;

numA = 1;
numD = numC[1-2-]; // numD = {0, 3, 4}
numB = numD[1];     // numB = 3
cooA = (numB, numA); // (3,1)
cooA = (yof(cooA), xof(cooA)); // (1,3)
display(cooA);
display(numA+numB);
```

Results:
(1,3)
4.0
Examples (2/4)

- Static image

```java
image imgA, imgB, imgC, imgD;

imgA = "sshield.jpg"\`(100,100);
imgB = imgA\`(800,100) @ 90;
imgC = imgA\`(800,500) @ 180 : -100;
imgD = imgA\`(100,500) @ 270 : -50;

display(imgA $ imgB $ imgC $ imgD);
```
Examples (3/4)

- Animated image

```javascript
defunc rotate_animation(image src, number time, number rotate_amount) image target
{
    target = animate(src @ rotate_amount * time, time);
}
image imgA = "strawberry.jpg";
coord cooA;
number time = 8, rotate = 360;

cooA = (500,300);
imgA = imgA ` cooA;
imgA = rotate_animation(imgA, time, rotate);
display(imgA);
```
Examples (4/4)

- GIF animation

```java
image dis;
image[] imgTar, imgSrc =
number counter = 0;

while(counter < 20) {
    imgTar[0] = imgSrc;
    counter = counter + 1;
}
dis = animate(imgTar, 10);
dis = dis` (900, 430)^5;
display(dis);
```
Lessons learned

- Things learned from Software Engineering actually works!
- Still, Team management.
  - Especially for Time management
  - Hard to find implement together!
- Need more fair distribution to learn
  - To learn something, everybody should do every procedure together that we have.
- Clarify how compiler works!
Q & A?

Thank you for listening our presentation