Photogram

Presented by:
Ohan Oda, Neesha Subramaniam,
Richard J Ng, Seikwon Kim

Overview of the presentation

- Overview of language
- Tutorials and examples
- Architectural design
- Summary and Lessons learnt

Overview

- Introduction

- What is Photogram?
- Who uses Photogram?
- Why use Photogram?

Overview

Motivation

- Photoshop has limited capability
- Easy to write lines of codes than using complicated interface of Photoshop for complicated image processing tasks
  - Example: Photomontage
- Some of the tasks are time consuming using Photoshop

Overview

- Features

- Easy-to-use: Java-like-syntax
- Portable
- Powerful
- Expandable

Tutorial

How to use Photogram

- Compiler: PGCompiler
- How to compile a .pg program
  - C:\java PGCompiler PGSampleProgram.pg
  - C:\javac PGSampleProgram.java
- How to run a .pg program
  - C:\java PGSampleProgram

Examples

- Collage

Column Escalator Library

- Importing functions and globals from other .pg files.

Examples

- Image Processing

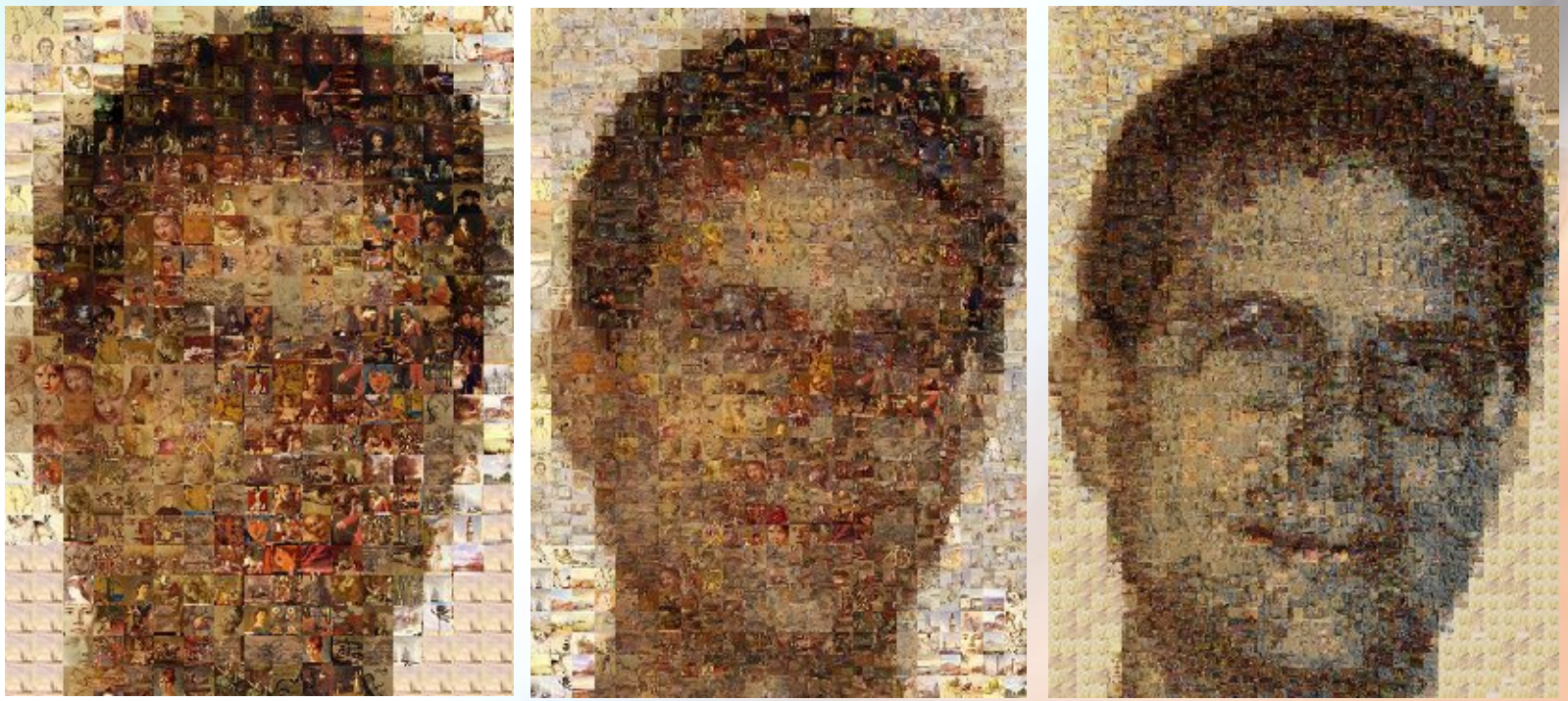
Examples

Optical Flow

- Used the library functions for drawing lines and circle in the output image.

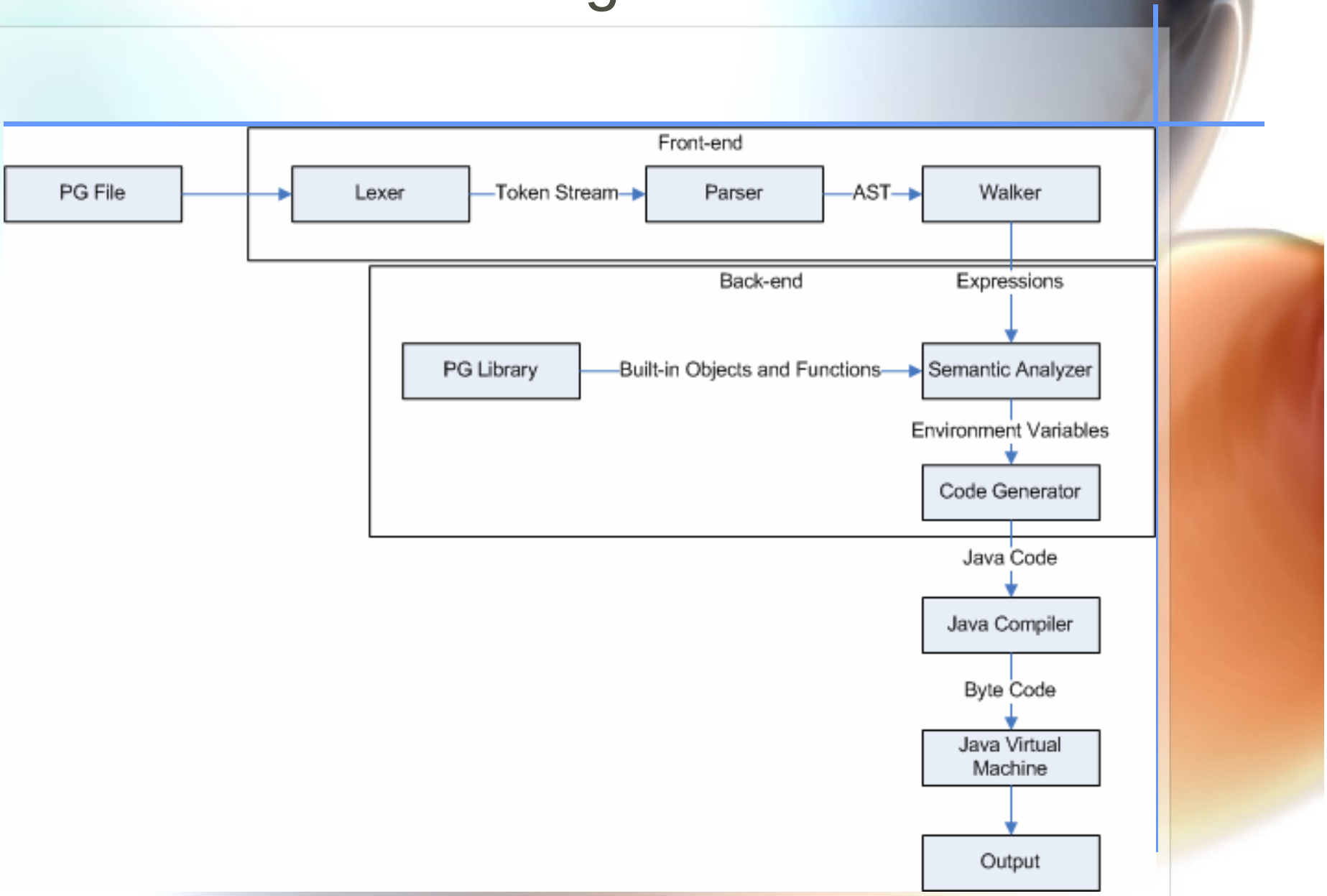
Examples

- PhotoMontage



- No. of images in database: 2500
- Algorithm: First match the average RGB of the image patch with the image in the database, then use cross-correlation to choose the best image.

Architectural design



Architectural design

Front-end

- Lexer
  - Produces a token stream
- Parser
  - Creates AST
- Walker
  - Calls Semantic Analyzer for Semantic Error Analysis

Architectural design

Back-end

- PG Library
  - Library of Built-in Objects and Functions
- Semantic Analyzer
  - Builds Symbol Table, Scopes, etc.
- Code Generator
  - Converts the PG code into Java Code

Testing

- Phase I
  - Grammar
- Phase II
  - Grammar
  - Majority was Walker and Semantic Analyzer
- Phase III
  - Final Sample

Testing

- Grammar Testing
  - Consists of one long file
  - As development, test grows larger
  - Test whether it parses well

Testing

- Phase II (Semantic)
  - Consists of small test files
  - Assume knowing nothing
  - Most of possible programming

Testing

- Phase III (Final Checking)
  - Check whether an actual program runs
  - Sample codes

Lessons Learned

- Keeping in touch with group members is very essential for successful completion of the project.
- Don’t ignore professor’s advice regarding CVS.
- Use JBuilder for developing your Java code.
- Write small test programs for each stage.

References

- Zhou, Tiantian, Feng, Hanhua, Ra, Yong Man, Lee, Chang Woo. “Mx: A programming language for scientific computation.” http://www1.cs.columbia.edu/~sedwards/classes/2003/w4115/Mx.final.pdf, May 2003.
- Ritchie, Dennis M. *C Reference Manual*. Bell Telephone Laboratories, 1975.
- http://www.photomosaic.com/rt/fineart.htm