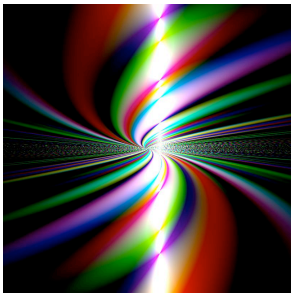
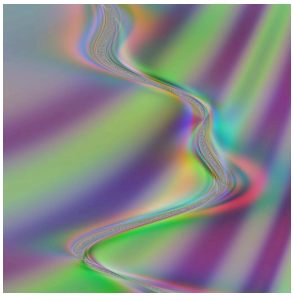
Objectives

- Final Project
- Analysis and Design
- Interpreting programming languages

1

Picasso Specification

- User can enter expressions
  - Interactively or from file
  - Language is defined in specification
- Many possible extensions


2

Project Deliverables Timeline

Deliverable	Who	Weight	Due Date
Preparation Analysis	Individual	10%	Fri, Oct 30
Preliminary Implementation	Team	30%	Mon, 11/9
Final Implementation	Team	45%	You decide →latest 11/19
Analysis	Individual	15%	Fri, Nov 20

Week 1: Understand code base, analyze/plan project
Week 2: Implement preliminary functionality
Weeks 3-4: Implement final version of application

3

ANALYSIS & DESIGN: FORMALIZED

4

Analysis Phase

- Create an abstract model in client's vocabulary
- Strategy:
  1. Identify classes that model (shape) system as set of abstractions
  2. Determine each class's purpose or main responsibility
     - member functions
     - data members
  3. Determine helper classes for each
     - Help complete responsibilities

Analysis Phase Discussion

- Expect to **iterate**
  - Won't find all classes at first
    - Especially helpers
  - Won't know all responsibilities
- Uncertainty in problem statement
  - May be concerns that need to be settled
  - Try to understand requested software system at level of those requesting software
- Rarely one true correct best design

Identification of Classes

- Potentially model the system
- Usually **nouns** from problem description or from domain knowledge
- Model real world/problem domain whenever possible
  - More understandable software
  - Helps during maintenance when someone unfamiliar with system must update/fix code

7

Identifying Responsibilities

- Responsibilities convey purpose of class, its role in system
- Questions to Ask:
  - What are the other responsibilities needed to model the solution?
    - Which class should take on this particular responsibility?
  - What classes help another class fulfill its responsibility?

8

Have You Modeled Everything?

- Strategy: Role playing
- Act as different classes: can you do everything you want in various scenarios?
  - Fill in missing classes, responsibilities
  - Methods: parameters, what returned
  - Restructure as necessary
    - No code yet so not actually refactoring
- Example **use cases**/scenarios:
  - User borrows a video and returns it two days late
  - User tries to borrow book that is already checked out

Definition of Use Case?

- Description of steps or actions between a user and a software system towards some goal

- What else can use cases be used for?
  - Test Cases!

TEAM FINAL PROJECT

11

Teams

BuildingBlocks	Abdul	August	Danny	Taylor	Will
Duplos	Bryan	Callie	Jacob	Leslie	
Linkimals	Dan	Jay	Nick	Sam	
MegaBlocks	Dominque	Jeremy	Katie	Raul	
JigsawPuzzles	Hayden	James	Laurie	Tara	

Teams, alphabetically by first name

Team toy analogy: individual pieces make a greater whole.

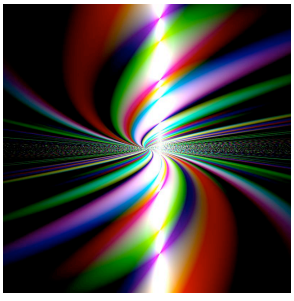
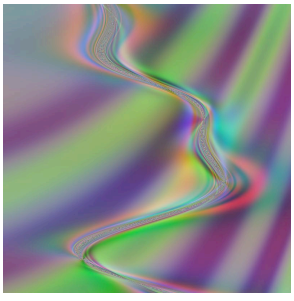
12

Project Metrics

- >1700 lines of code
  - Even more by the time your team is done
- Good for gaining experience
  - Large (for a course) piece of existing code that you need to build on
- Good for job interviews
  - Know the number of lines of code

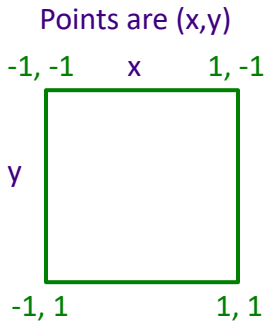
Picasso Specification

- User can enter expressions
  - Interactively or from file
  - Language is defined in specification
- Many possible extensions


Picasso Project Overview

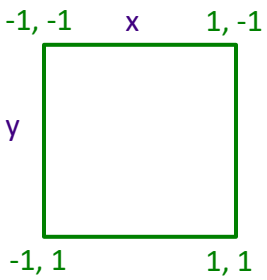
- Goal: Generate images from expressions
- Every pixel at position (x,y) gets assigned a color, computed from its x and y coordinate and the given expression
  - Range for x and y is [-1, 1]
- Colors are represented as RGB [red, green, blue] values
  - Component's range [-1, 1]
  - Black is [-1,-1,-1]
  - Red is [1,-1,-1]
  - Yellow is [1, 1,-1]



How is white represented?

Generating Images from Expressions

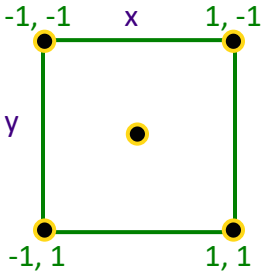
- ***Expressions*** at a specific (x,y) point/pixel evaluate to *RGB colors* [r,g,b]
  - `pixels[x][y] = expression.evaluate(x, y)`
- **x** evaluates to RGB color [x, x, x]
- In top right corner,
  - x evaluates to [1, 1, 1]
  - y evaluates to [-1, -1, -1]



Generating Images from Expressions

```
For all x:
    For all y:
        pixels[x][y] = expression.evaluate(x, y)
```

Consider evaluating expression as *f(x, y) = expression* at various points in the image



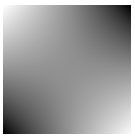
What is the resulting image if the expression is
- `[-1, 1, -1]` ?
- `x` ?
- `x*y` ?

17

Generated Images from Expressions

`[-1, 1, -1]`

`x`



`x*y`

```
For all x:
    For all y:
        pixels[x][y] = expression.evaluate(x, y)
```

If you click "Evaluate" in Picasso currently, it evaluates the expression `floor(y)`

18

Programming Language Syntax & Semantics

- What are the rules for an identifier in Java?
- What does an assignment statement look like in Java?
  - What can be on the left hand side?
  - What can be on the right hand side?
- What does a multiplication expression look like?
- How do we evaluate arithmetic expressions?

Programming Language Design

- Must be unambiguous
  - Programming Language defines a ***syntax*** and ***semantics***
- Interpreting programming languages
  1. Parse program into tokens
  2. Verify that tokens are in a valid form
  3. Generate executable code
  4. Execute code

Parsing into Tokens

- Example: x = 4*3; →

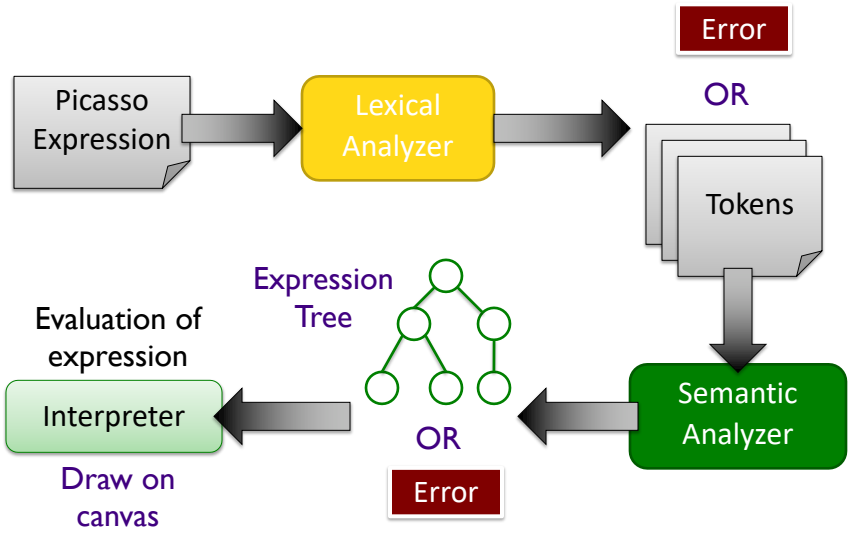
```
<id> <assignment> <num> <mult> <num> <endofstmt>
```

- Example: x = * 3 5;

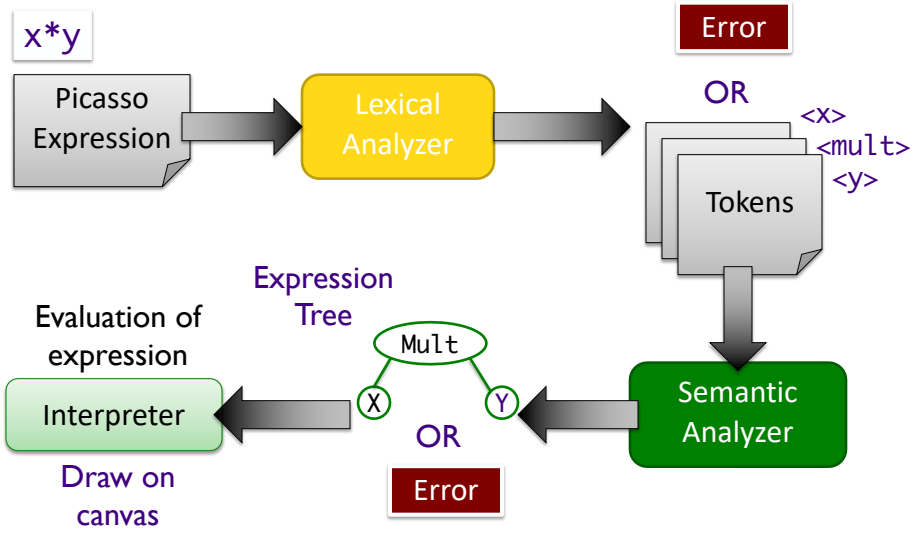
```
<id> <assignment> <mult> <num> <num> <endofstmt>
```

- Tokenizer doesn't care if statement is not valid
  - handled in next step
- Error example: 1x = 4**3;
  - 1x and ** are not valid tokens in Java

Interpreting the Picasso Language



Interpreting the Picasso Language



What We Need to Do/Represent

- Lexical Analysis
- Semantic Analysis
- Evaluation

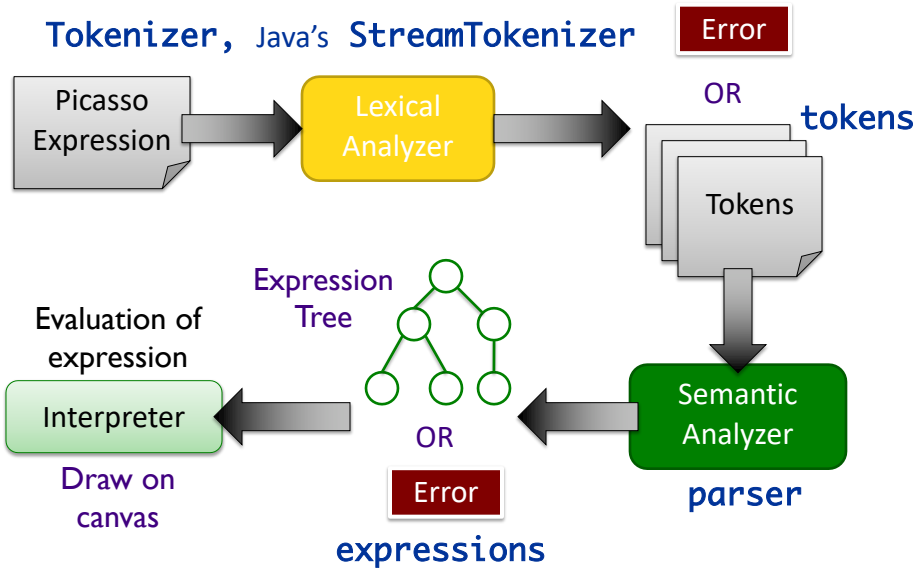
What We Need to Do/Represent

- Lexical Analysis
  - Recognize/create tokens
  - Report errors in creating tokens
- Semantic Analysis
  - Convert infix tokens into postfix
    - Report errors
  - Parse tokens into *expressions* (expression tree)
    - Report errors
- Evaluation
  - Evaluate expressions

Understanding the Code

- How does the given code map to lexical analysis, semantic analysis, and evaluation components?
  - Look for packages, classes that map to these steps

Interpreting the Picasso Language



Understanding the Code: Lexical Analysis

- Process
  - `picasso.parser.Tokenizer`
  - `picasso.parser.tokens.TokenFactory`
- Output:
  - `picasso.parser.tokens.*`

`FloorToken`

Understanding the Code: Semantic Analysis

- Process
  - picasso.parser.ExpressionTreeGenerator
  - picasso.parser.SemanticAnalyzer
  - picasso.parser.*Analyzer
- Output
  - picasso.parser.language.expressions.*

FloorAnalyzer

Understanding the Code: Evaluation

- Process
  - picasso.parser.language.ExpressionTreeNode
- Output:
  - RGBColor
- Displayed in PixMap on Canvas

Floor

Understanding the Code: Evaluation

- Key Parent class:
  `picasso.parser.language.ExpressionTreeNode`

```
public abstract RGBColor evaluate(double x, double y);
```

- "Old" version of expressions:
  - `ReferenceForExpressionEvaluations`

Understanding Code

- Run program
  - What does each button do?
- Start at Main.java
  - Follow calls to see where program goes
    - Breadth or depth-first search

This Week: Project Preparation

- Read over the Picasso (Final Project) specifications again
- 1st deliverable is a document that answers
  - What needs to be completed?
  - What is your plan for completing those tasks?
  - What tasks are you most interested in working on?
  - ….
- Friday
  - Discuss your plans, questions

TODO

- Project Analysis due Friday