Objectives

- Picasso Discussion
  - Best development practices
  - Singleton Design Pattern

1

Review

- What are the Picasso project components?
- What are the steps to add a new unary function into the Picasso language in the current implementation?
  - How much code needs to *change* to add the function?
- What can you do to help your team succeed?
- What is our work flow with Git?
- What is the spiral model of development?
  - For the preliminary implementation deadline, how would you categorize your prototype?

2

Review: Process of Adding Cosine Function to the Picasso Language

(in given code)

- Create a *token* for the cosine function
  - Same prefix as new function, e.g., `CosToken.java`
  - Needs to be added to `functions.conf`
- Create a *semantic analyzer* for the function with same prefix as function, e.g., `CosAnalyzer.java`
  - `Analyzer` class implements `SemanticAnalyzerInterface`, returns an instance of `ExpressionTreeNode`
- Create an `ExpressionTreeNode` for function: `Cosine.java`

Review: Teams Work Best When They are **Interdependent**

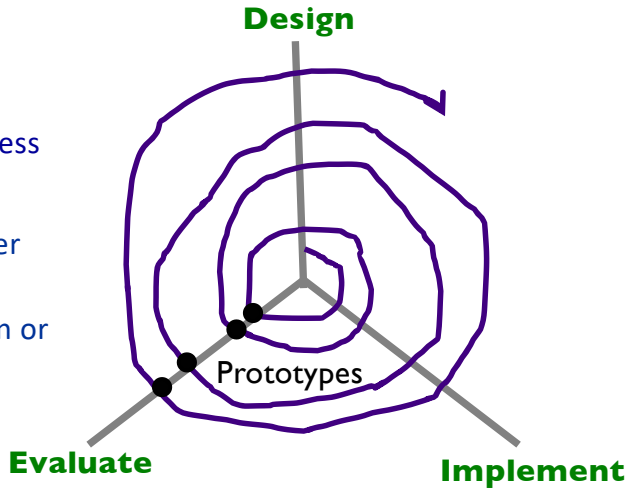
- In code terms, we want *loose coupling*
  - Depend on each other but don't depend on their details
- Consider
  - Are you allowing your team to truly be interdependent?
  - Who might be you be ignoring?
  - Who might be allowing themselves to feel inadequate?
  - How do you show appreciation for each other and yourself?

Review: Git WorkFlow

1. Create a branch from `main` for your work
   - Commit periodically
   - Write descriptive comments so your team members know what you did and why
2. Push your branch
3. On GitHub, open a ***Pull Request*** on your branch
   - Discuss and review potential changes – can still update
   - You can tag your teammates to let them know that you've completed your work
4. Merge pull request into `main` branch
5. In Eclipse, pull `main`

Review: Spiral Development Model

- Idea: smaller prototypes to test/fix/throw away
  - Finding problems early costs less
- In general…
  - Break functionality into smaller pieces
  - Implement most depended-on or highest-priority features first



[Boehm 86]

Radial dimension: cost

What Kind of Prototype is Picasso?

- Both for given code and for preliminary implementation
- High fidelity with respect to the GUI
- Depth
  - From GUI → Backend → GUI
  - But limited implementation of GUI features and Picasso language

7

SINGLETON DESIGN PATTERN

8

Problem: Too Many Objects!

- Sometimes, we only want one object to ***ever*** be created for a class
  - Often because there is some state that needs to be coordinated across the application

Solution: Singleton Design Pattern

- Make the constructor private
- Make a public method for accessing the one and only instance

Solution: Singleton Design Pattern

- Make the constructor private
- Make a public method for accessing the one and only instance (a static variable)

```
public class SemanticAnalyzer implements SemanticAnalyzerInterface {

    private static SemanticAnalyzer ourInstance;

    public static SemanticAnalyzer getInstance() {
        if (ourInstance == null) {
            ourInstance = new SemanticAnalyzer();
        }
        return ourInstance;
    }

    private SemanticAnalyzer() {
        …
    }

    public ExpressionTreeNode generateExpressionTree(Stack<Token> tokens)
```

Access to object

Private constructor

11

11

When Does Picasso Use the Singleton Design Pattern?

- Specialized analyzers need to refer to ***the*** `SemanticAnalyzer` to parse its parameters/ operators

```
return new Floor(
        SemanticAnalyzer.getInstance().
                generateExpressionTree(tokens) );
```

- Need to call methods on that one-and-only object

12

In Picasso:

Is the Singleton Design Pattern the Best Design?

- Is this the best design? <shrug/>
- Alternative 1: pass in the SemanticAnalyzer as another parameter:

```
public ExpressionTreeNode
  generateExpressionTree(Stack<Token> tokens,
       SemanticAnalyzer semAnalyzer);
```

- Alterative 2: make SemanticAnalyzer's methods be static
  - Requires making state static too

None of these changes are required; just explaining alternatives

Review: Generating Images from Expressions

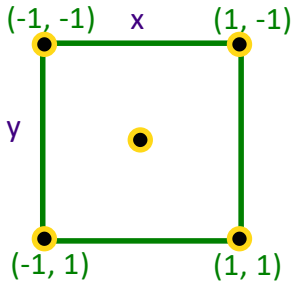
```
For all x:
    For all y:
        pixels[x][y] = expression.evaluate(x, y)
```

Consider evaluating expression as

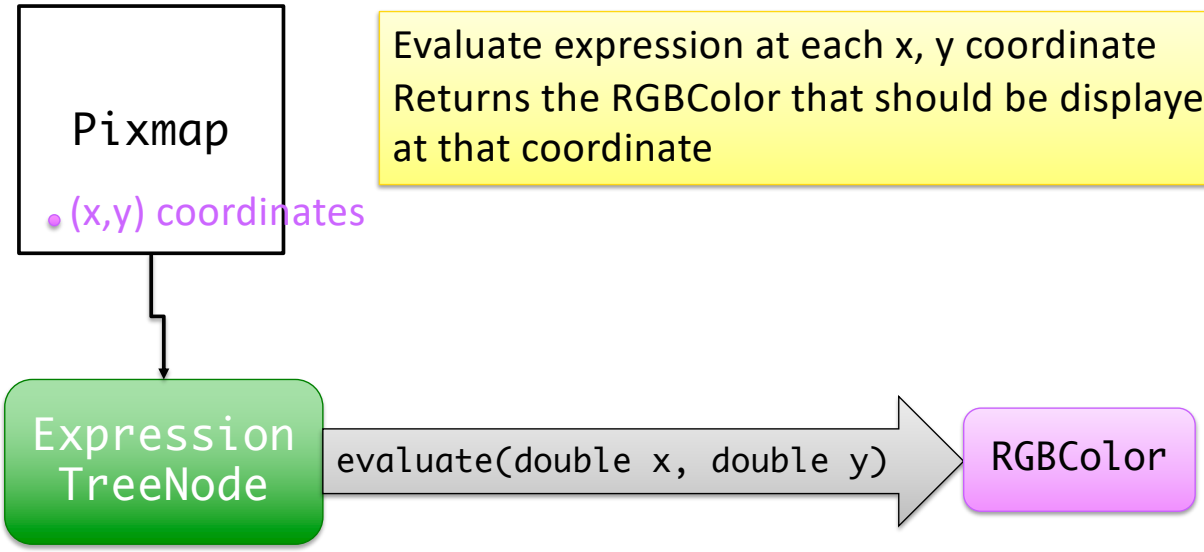
f(x, y) = expression

at various points in the image

Example: expression is x+y



Expression Evaluation



Evaluate expression at each x, y coordinate
Returns the RGBColor that should be displayed at that coordinate

15

Picasso Code: ReferenceForExpressionEvaluations

This implementation (from the "old" version of the code) is **different** from what we will have in our code. **But, it is a helpful reference.**

```
…
PLUS {
     public RGBColor evaluate(RGBColor left, RGBColor right) {
          double red = left.getRed() + right.getRed();
          double green = left.getGreen() + right.getGreen();
          double blue = left.getBlue() + right.getBlue();
          return new RGBColor(red, green, blue);
     }
},
…
```

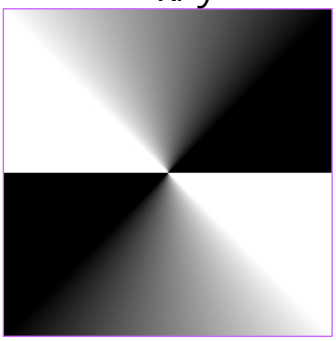
What are `left` and `right` referring to?

16

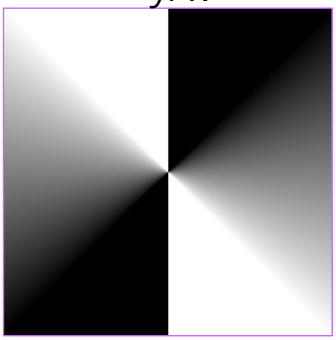
(placement of points is not exact in illustration)

x/y is not the same as y/x

x/y



y/x



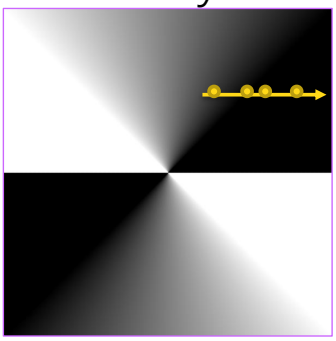
A common implementation mistake is the user enters x/y, but Picasso displays y/x.
Error may also be in x+y, but operation is commutative.

(placement of points is not exact in illustration)

x/y is not the same as y/x

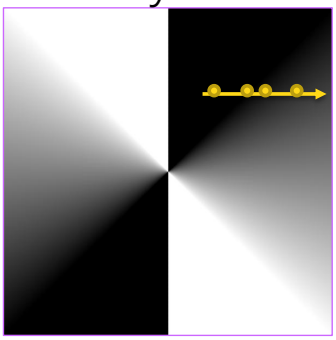
Consider points, holding y steady at -.5

x/y



Y	X	.3	.45	.55	.7
Y = -.5					
Color:					

y/x



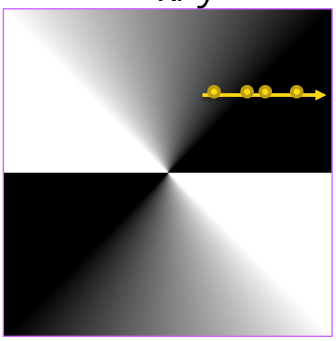
Y	X	.3	.45	.55	.7
Y = -.5					
Color:					

(placement of points is not exact in illustration)

x/y is not the same as y/x

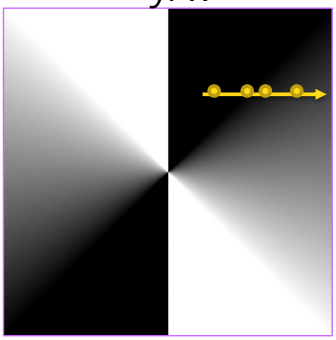
Consider points, holding y steady at -.5

x/y



Y	X	.3	.45	.55	.7
Y = -.5		-.6	-.9	-1.1	-1.4
Color:		Mid-gray	Dark gray	Black	Black

y/x



Y	X	.3	.45	.55	.7
Y = -.5		-1.67	-1.11	-.91	-.71
Color:		Black	Black	Dark gray	Mid dark gray

Team Collaboration/Planning

- An hour of thinking/design will save hours of coding

Preliminary Implementation

- Goals
  - Get your team working together
  - Find kinks in design
    - Rework now instead of later
- Tag your version
- Can keep working after that
  - Return to the tagged version for Friday's demo

Friday Demos: Preliminary Implementation

- Demo to me (only) in teams in Parmly 404
- Choose one person to demo the code
- Demo content:
  - Show what you have done for the preliminary implementation
  - Discuss design decisions
  - Tell me what you're thinking for extensions
- Order of teams will be randomly generated on Friday
  - Schedule: 8:35, 8:47, 9:00, 9:14
  - Schedule: 11:05, 11:17, 11:30, 11:44

Looking Ahead

- Friday: Preliminary Deadline and Demos
- Order of teams will be randomly generated on Friday
  - Schedule: 8:35, 8:47, 9:00, 9:14
  - Schedule: 11:05, 11:17, 11:30, 11:44
- Need to cancel today's office hours
  - Email with questions