

Objectives

- Analysis and Design
- Interpreting programming languages
- Final Project: Picasso

Nov 15, 2021

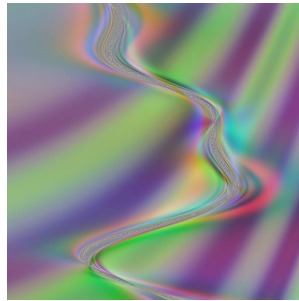
Sprenkle - CSCI209

1

1

Final Project: Picasso Specification

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



Nov 15, 2021

2

2

Project Deliverables Timeline

Deliverable	Who	Weight	Due Date
Preparation Analysis	Individual	10%	Fri, Nov 19
Preliminary Implementation	Team	15%	Fri, Dec 3
Intermediate Implementation	Team	15%	Fri, Dec 10
Final Implementation	Team	45%	You decide → latest 12/16
Analysis	Individual	15%	Fri, Dec 17

Before class

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

Nov 15, 2021

Sprenkle - CSCI209

3

3

ANALYSIS & DESIGN: FORMALIZED

Nov 15, 2021

Sprenkle - CSCI209

4

4

Analysis Phase

“Doohickey”

- 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

Nov 15, 2021

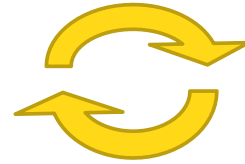
Sprenkle - CSCI209

5

5

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



Nov 15, 2021

Sprenkle - CSCI209

6

6

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

Nov 15, 2021

Sprenkle - CSCI209

7

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?

Nov 15, 2021

Sprenkle - CSCI209

8

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

Nov 15, 2021

Sprenkle - CSCI209

9

9

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!

Nov 15, 2021

Sprenkle - CSCI209

10

10

TEAM FINAL PROJECT

11

Teams

MastersOfThe Universe	Giorgio	Kailey	Liz	Maddie	Warren
Thundercats	Bennett	Danish	John	Nicholas	
Transformers	Billy	Jack	Jackson G	Peter	Taylor
Voltron	Dan	Dario	Jackson J	Patrick	

Teams, alphabetically by first name

12

Teams

Avengers	Ben	Caleb	Chapin	Garrett	Will
Eternals	Armando	Marshall	Mesoma	Nobel	
Justice League	Alex	Finn	Jorge	Ryan	
X-Men	Andrew	Jared	Praise	Sarah	Sujana

Teams, alphabetically by first name

Nov 15, 2021

Sprenkle - CSCI209

13

13

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

Nov 15, 2021

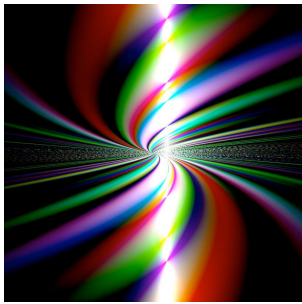
Sprenkle - CSCI209

14

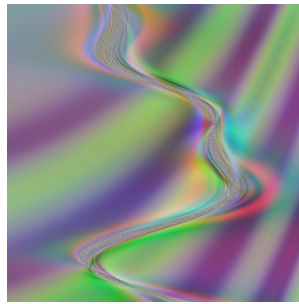
14

Final Project: Picasso Specification

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



Nov 15, 2021

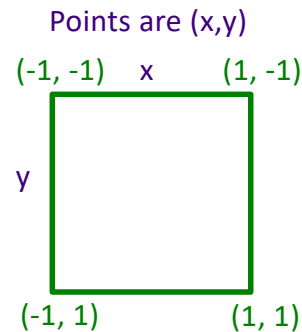


15

15

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
 - R, G, B component's range: $[-1, 1]$
 - Black is $[-1,-1,-1]$
 - Red is $[1,-1,-1]$
 - Yellow is $[1, 1,-1]$



How is white represented?

Nov 15, 2021

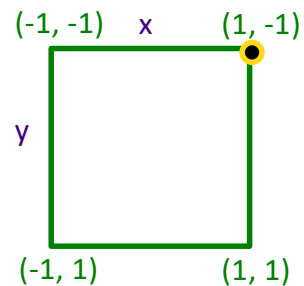
Sprenkle - CSCI209

16

16

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 $[-1, -1, -1]$



Nov 15, 2021

Sprenkle - CSCI209

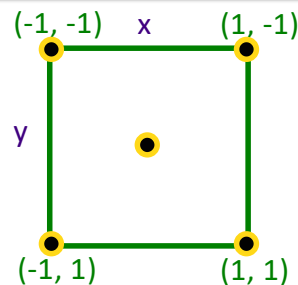
17

17

Generating Images from Expressions

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

Consider evaluating expression as
 $f(x, y) = \text{expression}$
 at various points in the image



Nov 15, 2021

Sprenkle - CSCI209

18

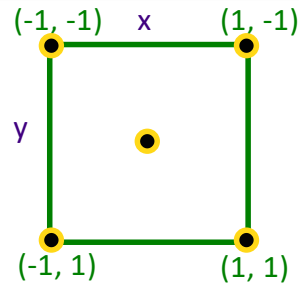
18

Generating Images from Expressions

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

Consider evaluating expression as
 $f(x, y) = \text{expression}$
 at various points in the image

Example: expression is $x+y$

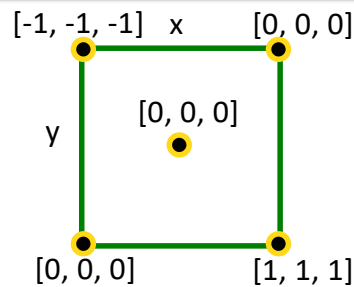


Generating Images from Expressions

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

Consider evaluating expression as
 $f(x, y) = \text{expression}$
 at various points in the image

Example: expression is $x+y$



Generating Images from Expressions

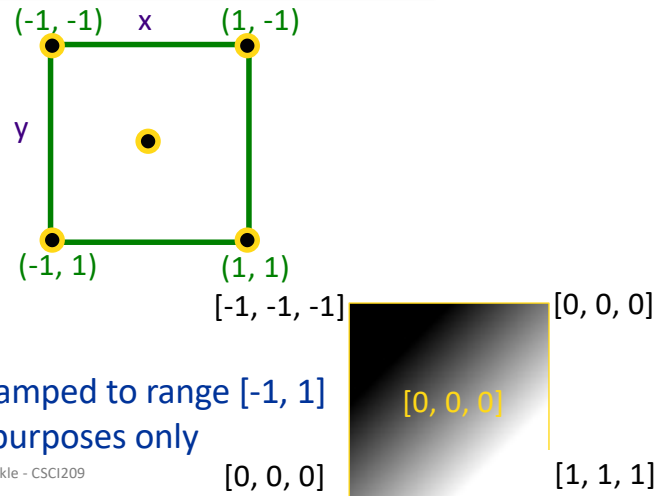
```
For all x:
  For all y:
    pixels[x][y] = expression.evaluate(x, y)
```

Consider evaluating expression as
 $f(x, y) = \text{expression}$
 at various points in the image

Example: expression is $x+y$

Resulting image:

- Recall that color range is clamped to range $[-1, 1]$
- Yellow outline for framing purposes only



Nov 15, 2021

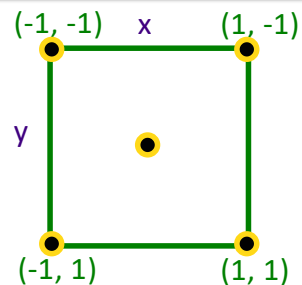
Sprenkle - CSCI209

21

Generating Images from Expressions

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

Consider evaluating expression as
 $f(x, y) = \text{expression}$
 at various points in the image



What is the resulting image if the *expression* is

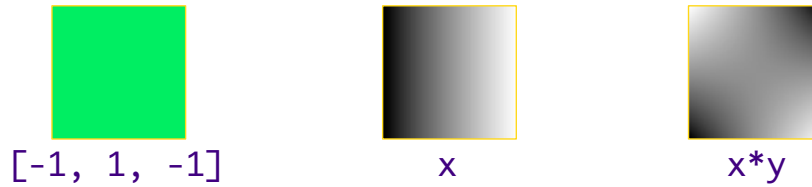
- $[-1, 1, -1]$?
- x ?
- $x*y$?

Nov 15, 2021

22

22

Generated Images from Expressions



```
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)`

Nov 15, 2021

Sprenkle - CSCI209

23

23

PROCESSING PROGRAMMING LANGUAGES

Nov 15, 2021

Sprenkle - CSCI209

24

24

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?

Nov 15, 2021

Sprenkle - CSCI209

25

25

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

Nov 15, 2021

Sprenkle - CSCI209

26

26

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

Nov 15, 2021

Sprenkle - CSCI209

27

27

Process of Understanding Code: Building Your Mental Model

- Apply spiral model to understanding code
- Review problem specification (low-cost effort)
- Explore project at the top-level (low-cost effort)
 - Look at packages, class names
 - Don't take a deep-dive until you have the bigger picture

<http://cs.wlu.edu/~sprenkle/cs209/projects/picasso/doc/>

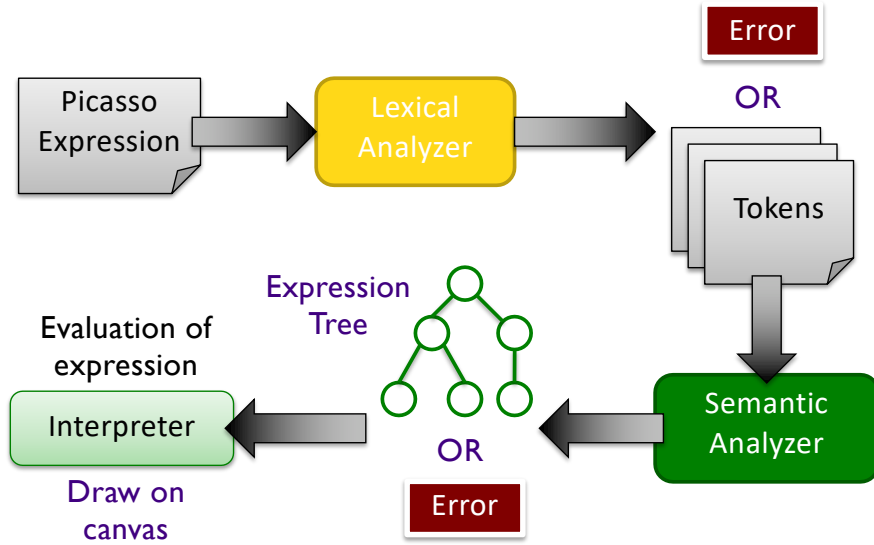
Nov 15, 2021

Sprenkle - CSCI209

28

28

Interpreting the Picasso Language



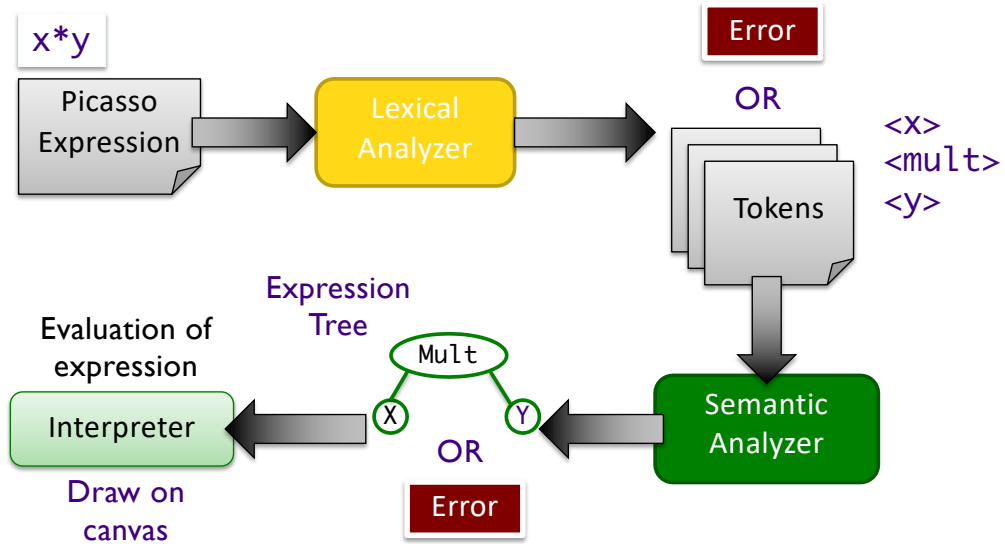
Nov 15, 2021

Sprenkle - CSCI209

29

29

Interpreting the Picasso Language



Nov 15, 2021

Sprenkle - CSCI209

30

30

What We Need to Do/Represent

- Lexical Analysis
- Semantic Analysis
- Evaluation

Nov 15, 2021

Sprenkle - CSCI209

31

31

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

Nov 15, 2021

Sprenkle - CSCI209

32

32

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

Nov 15, 2021

Sprenkle - CSCI209

33

33

Process of Understanding Code: Building Your Mental Model

- Look for important words/terms from problem domain
- Look for terms from design patterns
- Put code in black boxes or group code together

Oct 28, 2020

Sprenkle - CSCI209

34

34

Process of Understanding Code: Building Your Mental Model

- After you have the big picture, look at most important classes
- Decide: Does this class merit a closer look? Or do I just need the big picture of what it does?
 - Lean towards the latter towards the beginning
- Iterate!
 - Grow your mental model
 - What a “closer look” means changes over time
 - Early: what methods does the class have? What classes does this object interact with?
 - Later: what do these methods do? How does this class interact with other objects?

Oct 28, 2020

Sprenkle - CSCI209

35

35

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

Nov 15, 2021

Sprenkle - CSCI209

36

36

TODO

- Project Analysis due Friday

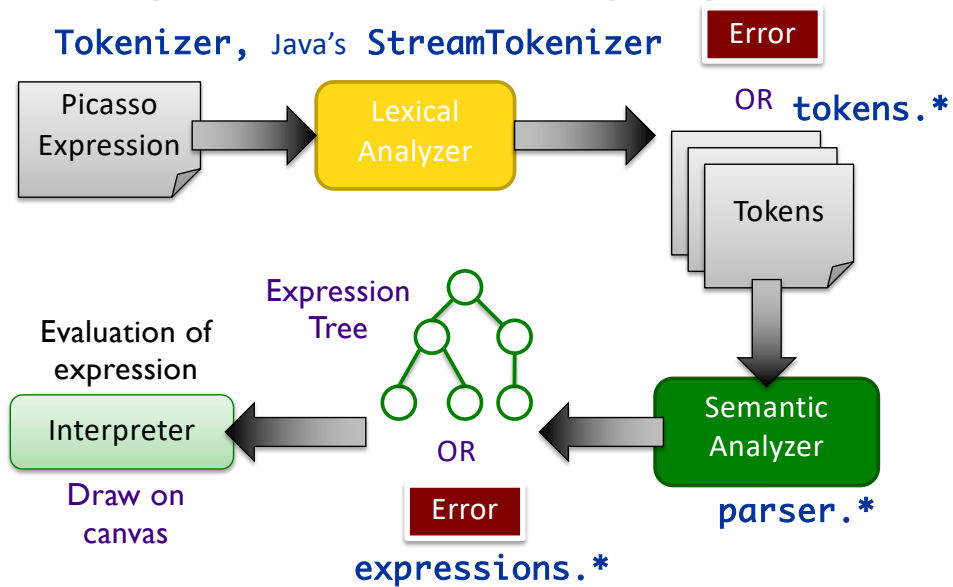
Nov 15, 2021

Sprenkle - CSCI209

37

37

Interpreting the Picasso Language



Nov 15, 2021

Sprenkle - CSCI209

38

38

Understanding the Code: Lexical Analysis

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

Nov 15, 2021

Sprenkle - CSCI209

FloorToken

39

39

Understanding the Code: Semantic Analysis

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

Nov 15, 2021

Sprenkle - CSCI209

FloorAnalyzer

40

40

Understanding the Code: Evaluation

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

Nov 15, 2021

Sprenkle - CSCI209

Floor

41

41

Understanding the Code: Evaluation

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

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

- “Old” version of expressions:
 - `ReferenceForExpressionEvaluations`

Nov 15, 2021

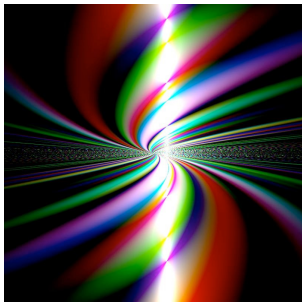
Sprenkle - CSCI209

42

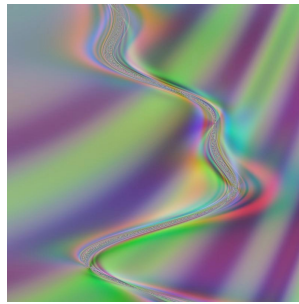
42

Final Project: Picasso Specification

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



Nov 15, 2021



43

43

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

Nov 15, 2021

Sprenkle - CSCI209

44

44

This Week: Project Preparation

- Read over the Picasso (Final Project) specifications
- 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