Uni Studies 3:
Intro to Processing

Assoc. Professor Donald J. Patterson
Uni Stu 3 Fall 2012
http://processing.org/
http://processing.org/
What the heck is Processing?
- A programming language
- An environment for running the programs

http://processing.org/
Intro to Processing

• What the heck is Processing?
  • A programming language
  • An environment for running the programs

• What is it for?
  • It is for people who want to create
    • images
    • animations
    • interactions

http://processing.org/
Intro to Processing

• What the heck is Processing?
  • A programming language
  • An environment for running the programs

• What is it for?
  • It is for people who want to create
    • images
    • animations
    • interactions

• Who is it for?
  • students
  • artists
  • designers
  • researchers
  • hobbyists

http://processing.org/
Intro to Processing

http://processing.org/
Intro to Processing

• Free to download

http://processing.org/
Intro to Processing

- Free to download
- Open source

http://processing.org/
Intro to Processing

- Free to download
- Open source
- Programs output in 2D, 3D or pdf

http://processing.org/
Intro to Processing

- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux

http://processing.org/
Intro to Processing

- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux
- Programs can be put in web pages

http://processing.org/
• Free to download
• Open source
• Programs output in 2D, 3D or pdf
• For Windows, Mac, Linux
• Programs can be put in web pages
• Programs can be run as applications

http://processing.org/
Intro to Processing

- Free to download
- Open source
- Programs output in 2D, 3D or pdf
- For Windows, Mac, Linux
- Programs can be put in web pages
- Programs can be run as applications
- Lots of documentation and books available

http://processing.org/
Excel → Low Complexity
Low Flexibility

Processing → Medium Complexity
High Flexibility

Computer Code → High Complexity
High Flexibility
Intro to Processing

http://processing.org/
Intro to Processing

- Your assignment for next week
  - Complete the lab:
    - “Getting Started. Welcome to Processing”

http://processing.org/
• Your assignment for next week
  • Complete the lab:
    • “Getting Started. Welcome to Processing”

http://processing.org/
Intro to Processing

• Your assignment for next week
  • Complete the lab:
    • “Getting Started. Welcome to Processing”

http://processing.org/
Your assignment for next week
• Complete the lab:
  • “Getting Started. Welcome to Processing”

11/21
Due:
• Complete Processing Exercise 1
• Turn in a picture of your sketch
In Class:
• Intro to next exercise

http://processing.org/
Intro to Processing

- Download the software v 2.0 beta 6

http://processing.org/
Intro to Processing

- Download the software v 2.0 beta 6

http://processing.org/
Intro to Processing

- Download the software v 2.0 beta 6

http://processing.org/
Download the software v 2.0 beta 6

http://processing.org/
Intro to Processing

• Run the software

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/
Intro to Processing

• Run the software

http://processing.org/

• This is the sketch window
Intro to Processing

• Run the software

• This is the sketch window
  • It is part of the Processing Development Environment (PDE)

http://processing.org/
Intro to Processing

- Run the software

- This is the sketch window
  - It is part of the Processing Development Environment (PDE)
  - This is where you put your program’s instructions

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/

- This is the Text Editor
Intro to Processing

- Run the software

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/

- This is the toolbar

Wednesday, November 14, 12
Intro to Processing

• Run the software

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/

- This is the message area
Intro to Processing

- Run the software

http://processing.org/
Intro to Processing

- Run the software

- This is the run and stop buttons

http://processing.org/
Intro to Processing

• Run the software

http://processing.org/

• To run a program
  • type in the program
  • hit run
  • look for the display window
Intro to Processing

• Run the software

To run a program
• type in the program
• hit run
• look for the display window

http://processing.org/
Intro to Processing

- Run the software

http://processing.org/

- To run a program
  - type in the program
  - hit run
  - look for the display window
Intro to Processing

- Run the software

To run a program
- type in the program
- hit run
- look for the display window

http://processing.org/
Intro to Processing

- What if something goes wrong?
What if something goes wrong?

An error will show up in the message area.
Intro to Processing

• What if something goes wrong?

• An error will show up in the message area

• Sometimes it will give you a clue about the problem
Intro to Processing

- What if something goes wrong?
  - An error will show up in the message area
  - Sometimes it will give you a clue about the problem
  - Make sure you are using parentheses in pairs
Intro to Processing

- What if something goes wrong?

  - An error will show up in the message area
  - Sometimes it will give you a clue about the problem
  - Make sure you are using parentheses in pairs
  - Make sure you end a line with a semi-colon
• What if something goes wrong?

• An error will show up in the message area
• Sometimes it will give you a clue about the problem
• Make sure you are using parentheses in pairs
• Make sure you end a line with a semi-colon
• Make sure you have the right number of parameters for your function
void setup()

void draw()
Intro to Processing

- Program flow

```
void setup()

void draw()
```
Intro to Processing

- Program flow

You can write collections of commands that get run in particular ways by Processing.
• Program flow

• You can write collections of commands that get run in particular ways by Processing

• the `setup` function is run once at the beginning
• Program flow

You can write collections of commands that get run in particular ways by Processing:

- the `setup` function is run once at the beginning
- the `draw` function is run repeatedly until the user hits stop
void setup() {
  size(480, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
• Program Flow

```java
void setup() {
  size(400, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 50, 50);
}
```
Intro to Processing

- Program Flow

- functions use curly braces to hold all the commands

```java
void setup() {
  size(400, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 88, 88);
}
```
Intro to Processing

- Program Flow

- functions use curly braces to hold all the commands

- `size()` changes the display window size

```java
void setup() {
  size(480, 128);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 88, 88);
}
```
Intro to Processing

• Program Flow

- functions use curly braces to hold all the commands
- `size()` changes the display window size
- `mousePressed` is true if the user is pressing the mouse button

```java
void setup() {
  size(400, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 88, 88);
}
```
Intro to Processing

• Program Flow

- Functions use curly braces to hold all the commands
- `size()` changes the display window size
- `mousePressed` is true if the user is pressing the mouse button
- `mouseX` and `mouseY` is the position of the mouse at the current time
Intro to Processing

- Program Flow

- functions use curly braces to hold all the commands
- size() changes the display window size
- mousePressed is true if the user is pressing the mouse button
- mouseX and mouseY is the position of the mouse at the current time
- fill() changes the color inside the shape that gets drawn next
void setup() {
  size(400, 120);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 30, 30);
}
void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
Intro to Processing

- Options

```java
void setup() {
  size(480, 480);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
Intro to Processing

• Options

- Menu bar shows shortcuts instead of requiring the buttons to be used
```java
void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```
Options

```java
void setup() {
  size(400, 400);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 10, 10);
}
```
Intro to Processing

- Options

- Create a new sketch
void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
• Options

```java
void setup() {
  size(480, 480);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
• Options

```java
void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
```

• Save the current sketch
Intro to Processing

```java
void setup() {
  size(480, 480);
}

void draw() {  // Run
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
Intro to Processing

- Options

```java
void setup() {
  size(480, 480);
}
void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
• Options

void setup() {
  size(480, 480);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}

• Open a previously saved sketch
void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}
Intro to Processing

• Options

```java
void setup() {
  size(480, 480);
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(128);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```
• Options

void setup() {
    size(480, 480);
}

void draw() {
    if (mousePressed) {
        fill(0);
    } else {
        fill(128);
    }
    ellipse(mouseX, mouseY, 80, 80);
}

• Share a sketch as an application
Intro to Processing
Intro to Processing

- Options
Intro to Processing

- Options

- Creates an application that you can share (or turn in)
Language API. The Processing language has been designed to facilitate the creation of sophisticated visual and conceptual structures.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Shape</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>() (parentheses)</td>
<td>createShape()</td>
<td>setstroke()</td>
</tr>
<tr>
<td>. (dot)</td>
<td>PShape</td>
<td>fill()</td>
</tr>
<tr>
<td>(def)</td>
<td>record()</td>
<td>noFill()</td>
</tr>
<tr>
<td><em>/</em>* (multiline comment)</td>
<td>ellipse()</td>
<td>stroke()</td>
</tr>
<tr>
<td>// (comment)</td>
<td>line()</td>
<td>strokeColor()</td>
</tr>
<tr>
<td>// (comment)</td>
<td>vertex()</td>
<td>strokeWeight()</td>
</tr>
<tr>
<td>= (assign)</td>
<td>rect()</td>
<td>strokeCap()</td>
</tr>
<tr>
<td>(string, access)</td>
<td>quad()</td>
<td>strokeJoin()</td>
</tr>
<tr>
<td>() (empty parentheses)</td>
<td>triangle()</td>
<td></td>
</tr>
<tr>
<td>catch</td>
<td>curve()</td>
<td></td>
</tr>
<tr>
<td>class</td>
<td>curveVertex()</td>
<td></td>
</tr>
<tr>
<td>cread ()</td>
<td>draw()</td>
<td></td>
</tr>
<tr>
<td>cxt()</td>
<td>endDraw()</td>
<td></td>
</tr>
<tr>
<td>extends</td>
<td>endSelection()</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>fill()</td>
<td></td>
</tr>
<tr>
<td>final</td>
<td>noFill()</td>
<td></td>
</tr>
<tr>
<td>implements</td>
<td>noStroke()</td>
<td></td>
</tr>
<tr>
<td>import</td>
<td>setstroke()</td>
<td></td>
</tr>
<tr>
<td>loop</td>
<td>strokeColor()</td>
<td></td>
</tr>
<tr>
<td>now</td>
<td>strokeWeight()</td>
<td></td>
</tr>
<tr>
<td>nul</td>
<td>text()</td>
<td></td>
</tr>
<tr>
<td>popStyle()</td>
<td>triangle()</td>
<td></td>
</tr>
<tr>
<td>private</td>
<td>translate()</td>
<td></td>
</tr>
<tr>
<td>public</td>
<td>vertex()</td>
<td></td>
</tr>
<tr>
<td>pushMatrix()</td>
<td>color()</td>
<td></td>
</tr>
<tr>
<td>rect()</td>
<td>noStroke()</td>
<td></td>
</tr>
<tr>
<td>return</td>
<td>strokeStyle()</td>
<td></td>
</tr>
<tr>
<td>setup</td>
<td>strokeWeight()</td>
<td></td>
</tr>
<tr>
<td>stroke()</td>
<td>text()</td>
<td></td>
</tr>
<tr>
<td>strokeWeight()</td>
<td>textFont()</td>
<td></td>
</tr>
</tbody>
</table>

2D Primitives:
- arc(): draw an arc
- ellipse(): draw an ellipse
- line(): draw a line
- rect(): draw a rectangle
- triangle(): draw a triangle

Curves:
- bezierDraw(): draw a smooth curve
- bezierPoint(): get a point on a curve
- arc(): draw an arc
- curveDraw(): draw a smooth curve
- curvePoint(): get a point on a curve
- curveTangent(): get the tangent at a point

3D Primitives:
- box(): draw a box
- quad(): draw a quad
- sphereDraw(): draw a sphere

Attributes:
- globalAlpha(): set the global alpha value
- noSmooth(): disable smooth shading
- smooth(): enable smooth shading
- strokeCap(): set the line cap
- strokeJoin(): set the line join
- strokeWeight(): set the stroke weight
What else can you draw?

Language (API). The Processing Language has been designed to facilitate the creation of sophisticated visual and conceptual structures.
Intro to Processing

• What else can you draw?

• Click on the link to get an example