Review

• Spatial Filters
  – Smooth
  – Blur – Low Pass Filter
  – Sharpen – High Pass Filter
  – Edge detection
  – Erosion
  – Dilation
• Other Pixel Filters
  – Thresholding
  – Posterize
  – Histogram Equalization
• Image Processing Applications

Ted Talk on Image Processing (thanks Leslie!)

Wearable projector and augmented reality
http://www.youtube.com/watch?v=nZ-VjUKAjas

What’s a string?

Characters enclosed by double quotes

"this is a String"
  "  this String starts with spaces"
"12345"
"the above String is made up of digit characters"

Print Strings to the Console using println()

println( "The mouse was pressed" );

Strings are Objects

Defined using a class
Have fields, methods, one or more constructors

String objects hold an array of 'chars'.

What’s a char?

  • A character enclosed by single quotes (’A’)

String msg = "I Love CS 110!";

Making Strings

• Declaring String objects with no chars
  String myName;
  String myName = new String();

• Declaring String objects init'd w/ char array
  String myName = "Dianna";
  String myName = new String("Dianna");

Chars are encoded by bytes

ASCII
  • American Standard Code for Information Interchange
  • An early character encoding standard
  • glyph <-> byte mapping
  • 127 characters
  • Forms the basis of new encoding standards
  • Unicode: more than 109,000 characters covering 93 scripts

Note:
  • Numbers are different than the digit characters
  • Includes special characters and punctuation
### String class methods

- **charAt(index)**  
  - Returns the character at the specified index

- **equals(anotherString)**  
  - Compares a string to a specified object

- **equalsIgnoreCase(anotherString)**  
  - S/A ignoring case (i.e. 'A' == 'a')

- **indexOf(char)**  
  - Returns the index value of the first occurrence of a character within the input string

- **length()**  
  - Returns the number of characters in the input string

- **substring(startIndex, endIndex)**  
  - Returns a new string that is part of the input string

- **toLowerCase()**  
  - Converts all the characters to lower case

- **toUpperCase()**  
  - Converts all the characters to upper case

- **concat(anotherString)**  
  - Concatenates String with anotherString

### Other forms of `indexOf()`

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| int     | `indexOf(int ch)`  
  - Returns the index within this string of the first occurrence of the specified character. |

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| int     | `indexOf(int ch, int fromIndex)`  
  - Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index. |

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| int     | `indexOf(String str)`  
  - Returns the index within this string of the first occurrence of the specified substring. |

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| int     | `indexOf(String str, int fromIndex)`  
  - Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |

### Other forms of `substring()`

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| String  | `substring(int beginIndex)`  
  - Returns a new string that is a substring of this string. |

<table>
<thead>
<tr>
<th>Returns</th>
<th>Description</th>
</tr>
</thead>
</table>
| String  | `substring(int beginIndex, int endIndex)`  
  - Returns a new string that is a substring of this string. |
Digit chars in a String are not integers

```java
String s = "12345";
void setup() {
    char myChar = s.charAt(1);
    byte myByte = byte(myChar);
    println(myByte);
}
```

Building Strings – Use ‘+’

```java
void setup() {
    String s1 = "Hello";
    String s2 = "World";
    String s3 = one + " " + two;
    println(s3);
}
```

```java
void setup() {
    String s1 = "She is number ";
    String s2 = " in computer science.";
    String s3 = s1 + 1 + s2;
    println(s3);
}
```

Special chars in a String using escape char(\)

Use the escape character to embed special characters in a String

```java
'\n' new line
'\t' tab
```

```java
void setup() {
    println("This is line 1\nThis is line 2");
}
```

Strings can be held by Arrays

– (Just like any other object or primitive type)

```java
String[] tokens = new String[] {"one", "two", "three", "four", "five"};
void setup() {
    println(tokens);
}
```

```java
String[] tokens = new String[5];
void setup() {
    println(tokens);
}
```

Strings can be held by Arrays

– Initialized when declared

```java
String[] tokens = new String[] {"one", "two", "three", "four", "five"};
void setup() {
    println(tokens);
}
```

Strings can be held by Arrays

– Not initialized

```java
String[] tokens = new String[5];
void setup() {
    println(tokens);
}
```
Built-in String functions (not methods)

split( bigString, splitChar )
• Breaks a String into a String Array, splitting on splitChar
• Returns new String Array

splitTokens( bigString, splitCharString )
• Breaks a String into a String Array, splitting on any char in splitCharString

join( stringArray, joinChar )
• Builds a new String by concatenating all Strings in stringArray, placing joinChar between each
• Inverse of split() function

nf( intValue, digits )
• Formats a number as a String

trim( theString )
• Removes whitespace from the beginning and end of theString

text( theString, x, y )
• Draws theString on the sketch at (x, y)

Split a String based on a single or multiple separator chars

String s1 = "12, 34, 56";
String[] as;
void setup() {
as = split(s1, " ");
//as = trim(as);
println( as );
}

String s1 = "Data: 12, 34, 56";
String[] as;
void setup() {
as = splitTokens(s1, ",:");
//as = trim(as);
println( as );
}

Join a String Array with a join char

String[] as = new String[] {"one", "two", "buckle my shoe"};
void setup() {
String s1 = join( as, " | ");
println( s1 );
}

one | two | buckle my shoe

Numbers can be formatted as Strings

phrase = s1 + nf(7, 3) + " " + s2;
// nf( integer, number of digits )
// "She is the 007 programmer."

phrase = s1 + nf(3.14159,3, 2) + " " + s2;
// nf( float, digits before decimal, digits after decimal )
// "She is the 003.14 programmer."