Interactive Graphics – Part 2

In this lecture
- Visual computing concepts
  - Animation
  - A particle system
- Programming concepts
  - If statement
  - Loops
  - Arrays
- Processing concepts
  - Colour + transparency
  - More shapes: rect
  - Random

Particle Systems

Particle simulation, though fairly simple to implement can create some very interesting visual effects.


http://www.siggraph.org/education/materials/HyperGraph/animation/movies/genesisp.mpg

Particle Systems

The emergent behaviour of particles systems can also be used to simulate some very complex high level phenomena such as cloth, hair and fluids.

More Complex Particles

Particle Dreams, Karl Sims (Siggraph 1988)

Particle Dreams, Karl Sims (Siggraph 1988)
Particles in Movie VFX

Images © 2012 Twentieth Century Fox Film Corporation. All rights reserved.

Particle Systems in Processing

- Particle systems are extremely common in movie visual effects, games, interactive digital art
- The following are some INTERACTIVE PARTICLE examples in processing (most should run in your browser):
  - Many more in openprocessing.org, search for keyword „particle“:

Implementing a particle system

Simple steps ... Just done a lot (P.S. the computer does the work not you)

First code this

- Track position of a circle
- Move it each frame and redraw
- Bounce off walls

A small extension to this

- For all particles in a collection of particles
- Track position of a circle
- Move it each frame and redraw
- Bounce off walls

Summary of Processing To Date

Takeaway Messages From Last Lecture

- Event-Driven Program
  - Call back: setup | draw | mousePressed | mouseClicked
- Variables
  - Type: int | boolean
  - Declaring and assigning variables
  - State-variables: mouseX, mouseY
- Functions in processing
  - Defining functions
  - Calling functions
  - Drawing shapes
  - Size | color
- Image objects (Pimage)
  - Load from files and display

Summary of Processing To Date

Takeaway Messages From Last Lecture

- Event-Driven Program
  - Call back: setup | draw | mousePressed | mouseClicked
- Variables
  - Type: int | boolean
  - Declaring and assigning variables
  - State-variables: mouseX, mouseY
- Functions in processing
  - Defining functions
  - Calling functions
  - Drawing shapes
  - Size | color
- Image objects (Pimage)
  - Load from files and display

Python Equivalent

```python
def setup():
    background(255, 255, 255)
```

```python
def draw():
    ellipse(mouseX, mouseY, 100, 100)
```

```python
def keyPressed():
    background(255, 255, 255)
```
Conditional (if/else statement)
- Allows program to make a decision about which code to execute.
- Conditions should be placed inside round brackets
  - This will be some kind of comparison
  - Usually with one of relational operators <, >, ==, !=, <=, >= (see next slide)
- Statements will be placed inside curly brackets (like functions)
  - Any code that you want to execute if the condition is met

//EXAMPLE
if (mouseX > 200)
    background(255, 255, 255);
else
    background(0, 0, 0);

// PYTHON EQUIVALENT
if mouseX > width/2:
    background(255, 255, 255)
else:
    background(0, 0, 0)

Relational Operations in Processing

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>DESCRIPTION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Check if two values are equal</td>
<td>if (key == 'a')</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>if (width &gt; 400)</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>if (width &lt; 400)</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>if (mouseX &gt;= 400)</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>if (mouseX &lt;= pmouseX)</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td>if (mouseButton != LEFT)</td>
</tr>
</tbody>
</table>

Arrays
Collections of things of the same type
- Very often in programs, variables are grouped into chunks of homogeneous data.
- Large collections of data are usually stored in Arrays.
- Commonly used Arrays in Visual Computing
  - An image is a collection of pixels.
  - A video is a collection of images.
  - A polygon is a collection of vertices (and edges).
  - A 3D object is a collection of polygons.
- For graphical objects we need arrays of at least 100 elements for things to be recognizable. Most real-world objects have tens of thousands or more elements!
- But, when it comes to writing programs dealing with 10 elements... is not really that different from dealing with 1000000.

*homogeneous == of identical type
Basic Arrays

- Declaring arrays in processing:
  ```
  int[] numbers = new int[3];
  ```

- Accessing arrays (pretty much like c/c++/java):
  ```
  numbers[0] = 90;
  numbers[1] = 150;
  numbers[2] = 30;
  ```

- Alternatively:
  ```
  int a = numbers[0] + numbers[1]; // Sets variable a to 240
  int b = numbers[1] + numbers[2]; // Sets variable b to 180
  ```

**UNLIKE IN PYTHON...**

We need to **declare** what type of object/value will be held in the array. Before we use it we also need to give the array a size (how many such objects are in the array)

Processing stuff

Introduce a few more concepts in processing that will be required for the next assignment

- RGBA color (N.B. American spelling)
  - Three main values; typically RGB i.e. red, green blue
  - RGBA includes a fourth value called ALPHA i.e. the transparency
  - Generally processing has a color class
    ```
    color c; // declaring a color variable
    c = color(255, 0, 125, 100); // creating a color
    ```
  
  - Some functions take the three values as input
    ```
    background(255, 0, 125, 100); // this is a colour that is reddish-purple and quite transparent.
    ```

RGB values are between 0-255

- Example uses:
  - Setting the interior colour of shapes
  - Setting the colour of outlines of shapes

```
// set the outline colour to black
stroke(0, 0, 0);
// set the fill colour to blue
fill(0, 0, 255);
```
random( ... )

- Perfect numbers can be visually boring (and sometimes unrealistic)
  - Randomness adds some richness to a visual program
- It also provides us some more data without us having to do much more work
- In processing...
  - random( max ) returns float value between 0 and max
  - random( min, max) returns a float value between min and max

Example:
```plaintext
void draw()
{
  float distance_right = random(width);
  float distance_down = random(height);
  float circle_size = random(10, 100);
  ellipse(distance_right, distance_down, circle_size, circle_size);
}
```

TRY THIS: Can you randomize the colours of the circles?

rect ( ... )

- A simple shape function to draw a rectangle
  - You need to provide the XY position of the TOP-LEFT corner as well as the WIDTH and HEIGHT of the rectangle
  - e.g. rect(100, 100, 500, 500); draws a rectangle 500 x 500 pixels wide with its top-left corner at 100, 100

Example:
```plaintext
void draw()
{
  float distance_right = random(width);
  float distance_down = random(height);
  float circle_size = random(10, 100);
  ellipse(distance_right, distance_down, circle_size, circle_size);
}
```

TRY THIS: Can you randomize the colours of the circles?