Design Programming

DECO1012
Rob Saunders
Rob Saunders

web: http://web.arch.usyd.edu.au/~rob

e-mail: rob@arch.usyd.edu.au

office: Room 274, Wilkinson Building
Wrapping Up and Moving On...
Simulation

- Computers can be used to simulate almost anything: from simple machines to complex organisms.

- In fact, most things that we do with computers are based on some form of simulation:
  - Word processors, spreadsheets and databases are (sophisticated) extensions of a simulated typewriters, (paper) spreadsheets and filing systems
A cellular automaton (CA) is a self-operating system comprised of a grid of cells and rules stating how each cell behaves in relation to its neighbor.

CAs are intriguing because of their apparent simplicity in relation to the unexpected results they produce.

Two common forms of cellular automata are Wolfram’s 1D cellular automata and Conway’s “Game of Life”
Wolfram’s 1D CAs

- Rules express how the state of a single cell changes based on its previous state and those of its two neighbours.
int[] rules = { 1, 0, 1, 0, 0, 1, 0, 1 };  
int gen = 0; // Generation  
color on = color(255);  
color off = color(0);  

void setup() {  
  size(101, 401);  
  background(0);  
  set(width / 2, 0, on); // Set the top, middle pixel to white
}  

void draw() {  
  // If we haven't reached the bottom of the window  
  if (gen < height - 1) {  
    // then increment the generation by 1  
    gen++;  
  } else {  
    // otherwise move the contents up one row  
    copy(0, 1, width, height - 1, 0, 0, width, height - 1);  
  }  
  for (int i = 1; i < width - 1; i++) {  
    // Construct the index into the rules  
    int r = 0;  
    if (get(i - 1, gen - 1) == on) r += 4;  
    if (get(i, gen - 1) == on) r += 2;  
    if (get(i + 1, gen - 1) == on) r += 1;  
    // Apply the rule to set the current state  
    if (rules[r] == 1) set(i, gen, on); else set(i, gen, off);  
  }
}
Particle Systems

- A particle system, an array of particles that respond to the environment or to other particles, serves to simulate and render phenomena such as fire, smoke, and dust.
- Hollywood films and video game companies frequently employ particle systems to create realistic explosions and water effects.
- Particles are affected by forces and are typically used to simulate physical laws for generating motion.
Simulation of Motion

- At each frame, update the position, velocity and acceleration of particles
  - To simulate constant motion:
    - position = position + velocity
    - \( x += \text{vx}; \)
  - To simulate accelerating motion:
    - velocity = velocity + acceleration
    - \( \text{vx} += \text{ax}; \)
    - \( x += \text{vx}; \)
class Particle {
    float x, y; // The x- and y-coordinates
    float vx, vy; // The x- and y-velocities
    float radius; // Particle radius
    float gravity = 0.1;

    Particle(int xpos, int ypos, float velx, float vely, float r) {
        x = xpos;
        y = ypos;
        vx = velx;
        vy = vely;
        radius = r;
    }

    void update() {
        vy = vy + gravity;
        y += vy;
        x += vx;
    }

    void display() {
        ellipse(x, y, radius*2, radius*2);
    }
}
Particle p;

void setup() {
    size(100, 100);
    noStroke();
    smooth();
    p = new Particle(0, height, 2.2, -4.2, 20.0);
}

void draw() {
    fill(0, 12);
    rect(0, 0, width, height);
    fill(255);
    p.update();
    p.display();
}
class Particle {
    float x, y; // The x- and y-coordinates
    float vx, vy; // The x- and y-velocities
    float radius; // Particle radius
    float gravity = 0.1;
    float ox, oy;
    float age = 0;

    Particle(int xpos, int ypos, float velx, float vely, float r) {
        x = xpos; y = ypos;
        ox = xpos; oy = ypos;
        vx = velx; vy = vely;
        radius = r;
    }

    void update() {
        vy = vy + gravity;
        y += vy; x += vx;
        age += 1;
        if (age > 200) { x = ox; y = oy; }
    }

    void display() {
        ellipse(x, y, radius*2, radius*2);
    }
}
int numParticles = 200;
Particle[] p = new Particle[numParticles];

void setup() {
    size(100, 100);
    noStroke();
    smooth();
    for (int i = 0; i < p.length; i++) {
        float velX = random(-1, 1);
        float velY = -i;
        p[i] = new Particle(width/2, height/2, velX, velY, 5.0);
    }
}

void draw() {
    fill(0, 36);
    rect(0, 0, width, height);
    fill(255, 60);
    for (int i = 0; i < p.length; i++) {
        p[i].update();
        p[i].display();
    }
}
Simulation of Force

- Newton’s second law states that the sum of the forces acting on an object is equal to the object’s mass multiplied by the object’s acceleration:
  - force = mass * acceleration;
- This equation can be rearranged to solve for the acceleration:
  - acceleration = force / mass;
The physics of a spring is simple and versatile.

A spring’s force is inversely proportional to how “stiff” the spring is and how far it is stretched. The equation for calculating the force of a spring is:

\[ \text{springForce} = -\text{stiffness} \times \text{stretch}; \]

If we know the length and the rest length of a spring the equation can be re-written as:

\[ \text{springForce} = -\text{stiffness} \times (\text{length} - \text{restLength}); \]

And velocity can be calculated:

\[ \text{velocity} = \text{damping} \times (\text{velocity} + \text{springForce}); \]
Libraries
Libraries

- A library is a collection of code, with a well documented interface, that is designed to encapsulate code for a particular task.
- Libraries are often intended to be “black boxes” such that the details of the code inside them is hidden from a programmer using them.
- A library will have a published API (Application Programming Interfaces) listing the variables, functions and/or objects that the library provides.
Libraries in Processing

- Processing supports two types of library
  - Core libraries are supplied with Processing
    - The core libraries include Video, Net, Serial, OpenGL, PDF Export, DXF Export, XML Import, and Candy SVG Import and are documented on the Processing website.
  - Contributed libraries are written by members of the Processing community
    - The contributed libraries typically cover more specialised features, e.g., computer vision
Another Dimension

- Processing supports drawing in 3D either using the P3D renderer or by using the core OpenGL library
Hardcopy

- Processing supports drawing for print media as PDF documents
- You can use the same drawing commands as usual but you enclose them in `beginRecord()` and `endRecord()` statements
Sound and Music

- There are several contributed libraries for manipulating sound and music:
  - Minim, Ess, jm-Etude, Sonia

- Which library you use depends on what you want to do with the audio...
Vision

- The video library provided with Processing handles both the loading of video from a file and the processing of video from a camera...
- There are also libraries to make it really easy to do common tasks like “blob detection”
User Interfaces

• A very useful type of library for developing more complex sketches is a GUI library

• There are several contributed GUI libraries for Processing, the most complete of which is the ControlP5 library...
Physics

› The “Traer Physics” library provides an excellent basis for developing simulations
Beyond Processing
The Processing Family
Mobile Phones

- Mobile Processing is a sister project to Processing designed for mobile phones
- The interface and programming language are similar to Processing, although some of the supported features are different
  - http://mobile.processing.org/
- Why program for mobile phones?
  - Mobile phones are the most widespread technology in the world (3.3 billion in Nov. 2007)
Electronics

- Wiring and Arduino are two closely related projects that provide a Processing-like programming environment and hardware interface board.
- Both projects use the Wiring programming environment, which is based on Processing.
- The language is a bit different because it is based on C instead of Java.
- The I/O boards differ between Wiring and Arduino boards and come in a variety of types.
Other Programming Languages
Types of Languages

‣ Compiled versus Interpreted Languages
  ‣ Compiled languages are generally faster when run, interpreted languages are quicker to develop with

‣ Client-Side versus Server-Side Languages
  ‣ Client-side languages allow more interactivity with a user, server-side languages run on a known hardware platform

‣ Text-Based versus Visual Prog. Languages
Max/MSP/Jitter

```
<table>
<thead>
<tr>
<th>timepoint 5 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0.</td>
</tr>
<tr>
<td>metro 16n @quantize 16n @autostart 1</td>
</tr>
<tr>
<td>metro 4n @quantize 4n @autostart 1</td>
</tr>
<tr>
<td>transport</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0.</td>
</tr>
<tr>
<td>bars</td>
</tr>
</tbody>
</table>
```
Programming Languages

- General Purpose Languages
  - C/C++, C#, Java, JavaScript, Lisp, Objective-C, Pascal, Perl, PHP, Python, Ruby

- Application Scripting Languages
  - ActionScript, AppleScript, Max/MSP, MEL (Maya Embedded Language), RhinoScript
A Quick Review
What have we learned?

‣ We’ve covered a lot of ground in this unit of study, so what have we learned?
  ‣ Fundamentals of programming
    ‣ Instructions
    ‣ Variables
    ‣ Functions
    ‣ Objects
    ‣ Libraries
What can you do now?

‣ How can you use your newly acquired ninja programming skills?
  ‣ Use code to be more creative with computers
  ‣ Get computers to be (a bit more) creative?