

LCC 6310 The Computer as an Expressive Medium

Lecture 7

Overview

Programming concepts

Subclasses

Processing, pp.320-322

Methods that return values

Talk about Assignment 2

Project 1 - examples & discussion

Revisit our example

So far we have a rocket that flies around in a field of asteroids...

What if we want our rocket to be able to fire missiles?

But we don't want to get rid of our non-firing rocket

We can create a subclass!

Inheritance

Subclasses inherit fields and methods from parent

We create our subclass as follows:

```
class ArmedRocket extends Rocket {  
    ...  
}
```

Let's try to add this to our Rocket example in [Processing...](#)

Our subclass needs a constructor

Our empty ArmedRocket example causes an error

Processing doesn't know how to construct an ArmedRocket

We want the ArmedRocket constructor to do the same work as the Rocket constructor

```
ArmedRocket(int initialX, int initialY, float initialRot) {  
    super(initialX, initialY, initialRot);  
}
```

The keyword `super` means to call the method in the parent with the same name. Back to [Processing...](#)

Now we have ArmedRocket

We can now use an ArmedRocket in our example

Instead of creating an instance of Rocket, we can create an instance of Armed Rocket in [Processing...](#)

But at the moment it's basically just a copy of Rocket

The only reasons to define an ArmedRocket are:

- To add new capabilities, or
- To override old ones

So add a fire() method

An ArmedRocket should be able to fire missiles

So we want a fire method to draw a missile that shoots out of the rocket

We could have the fire method draw the missile...

Is there a problem with this?

Missiles should also be objects

The object oriented solution is to make the missile an object as well

All the different types of "things" in our domain should have a corresponding class

Like asteroids and rockets, the missile class should know how to draw itself

A Missile is similar to a rocket (position, rotation, draw method, etc.)

But its initial position will be the position of the ArmedRocket that fired it

Also, missiles aren't going to wrap around the display. Instead, they should travel on a straight path and disappear off-screen.

Let's create a missile class...

The missile class

```
class Missile {
    final float velocity = 150;
    float xPos, yPos;
    float velocityX, velocityY;
    long lastDrawMillis;

    Missile(float initialX, float initialY, float rotation) {
        xPos = initialX;
        yPos = initialY;

        // get the missile's velocity in x and y directions based on its orientation
        velocityX = sin(rotation) * velocity;
        velocityY = -cos(rotation) * velocity;

        lastDrawMillis = millis();
    }

    // Also needed: drawMe() and isVisible() methods...
}
```

Missile drawMe()

```
void drawMe() {
    // update the current and last draw times
    long currentMillis = millis();
    float timeSinceLastDraw = ((float)currentMillis - (float)lastDrawMillis)/1000;
    lastDrawMillis = currentMillis;

    // update the position based on the velocity and the time
    // elapsed since the last time the asteroid was drawn
    xPos = xPos + velocityX * timeSinceLastDraw;
    yPos = yPos + velocityY * timeSinceLastDraw;

    pushMatrix();
    translate(xPos, yPos);
    ellipse(0, 0, 5, 5);
    popMatrix();
}
```

When to draw the missile?

Missiles travel on a straight path, so we only want to draw a missile when it's within the bounds of the display

So we'll need a method that checks if the missile is within bounds

It returns a **boolean**: true if in bounds, false otherwise

```
boolean isVisible() {
    // missiles don't wrap around so they are only visible when
    // they are within the bounds of the display window
    if ((xPos > XSIZE) || (xPos < 0) || (yPos > YSIZE) || (yPos < 0))
        return false;
    else
        return true;
}
```

The fire() method

Now that we have a missile object, our `ArmedRocket.fire()` method can just create and return a missile...

```
Missile fire() {
    Missile m = new Missile(xPos, yPos, rotation);
    return m;
}
```

Now we need to add code in `draw()` to draw missiles...

But first, a place to store missiles

First we'll need some variables to store the missiles

We can use an array like we did for Asteroids

We'll also need a size for the array. This will be the maximum number of missiles that can be displayed at a time

```
static final int MAX_MISSILES = 5;
Missile[] m;
```

We also need to create an array instance in our setup() method

```
m = new Missile[MAX_MISSILES];
```

Now for the drawing part...

Drawing missiles

In our draw() method, we want to draw all the existing missiles that have been fired by our ArmedRocket

```
for(int i = 0; i < MAX_MISSILES; i++) {
  if (m[i] != null) {
    m[i].drawMe();
    if (!m[i].isVisible())
      // delete missile when it goes off-screen
      m[i] = null;
  }
}
```

That's all great, but how do we fire?!

Let's use the keystroke 'm' to fire a missile

```
if (key == 'm') {
  // Only add a missile if there are less than MAX_MISSILES on screen
  println("Pressing the missile button");
  for(int i = 0; i < MAX_MISSILES; i++) {
    if (m[i] == null) {
      m[i] = rl.fire();
      break; // java keyword to break out of a loop - no reason to keep
             // looking for blank missile slots once we've found one
    }
  }
}
```

Where do we add this code?

What happens if we put it where we currently check for other keystrokes?

Using the keyPressed() method

So far we've been checking for key presses in the draw() loop

If we fire missiles in the draw() loop, notice that we fire many times even if we push the key quickly

This is because draw() is called many times a second

Instead we can use the keyPressed() method

keyPressed() is a built in Processing method that is called when a key is pressed (similar to the mousePressed() method we've seen already)

Useful because you can tie events to keystrokes instead of to draw()

keyPressed() method

```
void keyPressed() {
  // With keyPressed(), we only check to see if we should fire a missile
  if (key == 'm') {
    // Only add a missile if there are less than MAX_MISSILES on screen
    // Runs through the missile array checking for any empty slots (null)
    for(int i = 0; i < MAX_MISSILES; i++) {
      if (m[i] == null) {
        m[i] = r1.fire();
        break; // java keyword to break out of a loop - no reason to keep
              // looking for blank missile slots once we've found one
      }
    }
  }
}
```

Much better!! Now our ArmedRocket works as desired.

Assignment 2

Posted online, due **Friday**

- A2-01: Using beginShape() and endShape(), create a composition with five or more vertices.
- A2-02: Using beginShape() and endShape(), create a composition with ten or more vertices.
- A2-03: Create an image different from A2-02, but using the same vertex data.
- A2-04: Write a function with one parameter and demonstrate it visually.
- A2-05: Write a function for drawing triangles and visually demonstrate its flexibility.
- A2-06: Write a function with three or more parameters and visually demonstrate its flexibility.
- A2-07: Create a dynamic animation using the cos() function as a generator for motion.
- A2-08: Create a dynamic animation using the cos() and sin() function as a generator for motion.
- A2-09: Move two visual elements across the screen using the random() function as a generator of movement. Give each element a unique nonlinear motion.
- A2-10: Create an event that begins when the mouse is pressed and ends when the mouse is released.
- A2-11: Create a responsive image that behaves differently when the mouse is moving and the mouse is dragging.
- A2-12: Create a button that changes the color of the background when it is clicked.
- A2-13: Program your moving elements from A2-09 but use classes to represent the two visual elements.
- A2-14: Create a subclass of one of the asteroids classes that adds a new capability. Some examples of what you could do: create a subclass of Rocket (or ArmedRocket) that shoots flame when the thrusters are fired and/or plays a sound when thrusters are fired, create a subclass of Asteroid that knows when it's been hit (instead of doing this test in loop()), create a subclass of Asteroid that splits into two smaller Asteroids when it's hit.

Project 1 discussion

Let's look at some of your project 1s...

Volunteers? (This is a good way to get some feedback from the whole class... if not I'll pick some...)

Things to think about

Concept: does the concept make sense? is it original? does it require explanation or is it easily grasped?

Temporal representation: is it easy to read the time? is the flow of time evident? does it show different time scales?

Visual presentation: does it have aesthetic appeal? fluidity? color coherence? what are the criteria of visual appeal? can they be objective?

Interaction: (wasn't required but some projects made use of interaction...) is it clear? responsiveness? mappings?

Taking apart the source code

Volunteers? (If not I'll pick some...)

Remember...

For **Thursday** this week: Theory Readings

Two presenters (you know who you are!)

Everyone else: prepare one discussion question for each reading

A Cyborg Manifesto: Science, Technology, and Socialist-Feminism in the Late Twentieth Century - Donna Haraway (NMR pp.515-542)

The GNU Manifesto - Richard Stallman (NMR pp.543-550)