The Imaginary Institute
www.imaginary-institute.com
2Dcourse@imaginary-institute.com

2D ANIMATION & INTERACTION COURSE
COURSE VIDEO SYLLABUS

WEEK 1: A STRONG FOUNDATION

We start by building a good foundation. We see how to install Processing, how to work with colors, and how to draw and control the appearance of some basic but useful shapes.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 1 Overview *(7m46s)*

Group 2: Installing and Using Processing. We start out by preparing for the journey, making sure we have the right attitude and expectations. Then we install Processing, take a tour of the interface, and make our first picture**.**

Processing Tour *(11m23s)*
Programming As A Medium *(4m18s)*
Installing Processing on Windows *(7m28s)*
Installing Processing on Mac *(7m0ss)*
The Processing Window Part 1 *(7m43s)*
The Processing Window Part 2 *(9m28s)*
Creating A Picture *(12m35s)*

Group 3: Color. Color is essential to everything we draw. Processing offers a couple of different ways to specify colors.

Color Systems Part 1 *(8m44s)*
Color Systems Part 2 *(5m50s)*
RGB *(8m21s)*
HSB *(8m53s)*
Color Pickers *(11m21s)*
Defining HSB Colors *(9m42s)*
Using Colors *(4m36s)*
Transparency *(9m51s)*
Grays *(3m45s)*

©The Imaginary Institute, 2013. All rights reserved.

Group 4: Basic Graphics. To draw something on the screen, we need to tell Processing where it's located, and what shape it has. In this group we see how to draw three important basic shapes: lines, rectangles (including squares), and ellipses (including circles).

Coordinates *(6m0s)*

fill and stroke *(11m55s)*

Style *(10m21s)*

Lines *(2m38s)*

Rectangles *(8m31s)*

Ellipses *(7m39s)*

Group 5: Putting it Together. We take everything from above and use it to write a *sketch*, or program, to draw a picture.

Putting It Together Part 1 *(8m17s)*

Putting It Together Part 2 *(7m45s)*

Group 6: Recap and Homework. We summarize this week's material so it's all in one place, and then it's time for you to put it into practice in your first homework assignment.

Week 1 Recap *(3m35s)*

Week 1 Homework *(8m55s)*

Group 7: Supplements. The videos in this section are optional supplements. They're here to illuminate interesting ideas, answer some questions you might have wondered about, and generally flesh out some of the topics we've seen.

Processing On Windows *(4m3s)*

Why 255 *(12m51s)*

Color Displays *(12m19s)*

Color Blending *(8m52s)*

Stroke Caps *(8m38s)*

WEEK 2: ANIMATION

This week we dig deeper into making images. We'll find that we can create our own named objects to hold numbers, and how that this gives us a huge amount of expressive power. In particular, we'll see how to use these objects, called *variables*, to produce animation. We'll see how to put all these pieces together in a program that we can run to create an animated piece.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 2 Overview *(6m11s)*

Group 2: Names. Our programs will contain many objects, and we'll give each one a name.

Names Part 1 *(7m1s)*

Names Part 2 *(8m19s)*

Group 3: Variables. A variable is a key element of every computer program: it's an object that can hold a value, such as a number. Once we assign a value to a variable, it holds that value until we assign it something new. Different types of variables are designed to hold different types of information. We'll meet the two most important numerical types, and see how to use them.

Integers Part 1 *(9m38s)*

Integers Part 2 *(6m43s)*

Using Integers *(12m55s)*

floats *(9m48s)*

Ints And Floats *(10m33s)*

Dividing Integers Part 1 *(7m16s)*

Dividing Integers Part 2 *(6m44s)*

Group 4: Animation. To create animation, we produce a series of still images one after the other. If each new frame is just a little bit different than the previous one, and new images are displayed quickly enough, the human visual system will interpret the sequence of still images as a moving picture.

Animation Part 1 *(7m9s)*

Animation Part 2 *(8m47s)*

frameCount *(9m38s)*

Group 5: Writing Programs. When our programs grow to more than a few lines long, it's useful to add some documentation, called a comment, right in the program to help us understand what's going on. If something goes wrong, we say that our program has a "bug." There are an endless variety of bugs, but there are a few standard techniques for tracking down the source of a bug so we can correct it.

Comments *(7m57s)*

Color Coding *(9m40s)*

Errors *(9m48s)*

Printing *(8m29s)*

Debugging *(7m8s)*

Group 6: Homework and Recap. We survey what we've seen this week, and then you get to put it into action in your homework.

Recap Week 2 Part 1 *(8m25s)*

Recap Week 2 Part 2 *(7m43s)*

Homework Week 2 *(8m4s)*

Group 7: Supplements. The videos in this section are optional. They're here to illuminate interesting ideas, answer some questions you might have wondered about, and generally flesh out some of the topics we've seen.

longs *(10m52s)*

Scientific Notation *(10m39s)*

Doubles *(9m45s)*

Preferences *(5m51s)*

External Editor *(10m53s)*

Tabs *(7m59s)*

WEEK 3: INTERACTION

Week 3 is about building up our programming skills and getting started on writing interactive animations. Now that you have solid control of the fundamentals, we'll start moving much faster to expand our expressive power and control.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together. Then we'll write a very little game. We'll do it pretty quickly and a lot of the steps will be new, but by the end of this week you'll understand everything we've done.

- Week 3 Overview *(6m31s)*
- A MiniGame Part 1 *(9m10s)*
- A MiniGame Part 2 *(7m9s)*
- A MiniGame Part 3 *(7m24s)*

Group 2: Variables. We'll expand our control of variables in two ways. First, we'll see how to create and use color variables so we don't have to type in numbers every time we change colors. Then we'll look at a number of universally-used shorthands for doing basic numerical operations with variables.

- Color Variables Part 1 *(9m11s)*
- Color Variables Part 2 *(9m50s)*
- Using Color Variables Part 1 *(7m18s)*
- Using Color Variables Part 2 *(7m49s)*
- Shorthand Arithmetic Part 1 *(12m36s)*
- Shorthand Arithmetic Part 2 *(4m54s)*

Group 3: Functions. We often want to package up bits of code so that we can repeat them in several places without having to retype everything. The mechanism of a *function* lets us do that. Even better, we can tell the function that some of its variables (called *arguments* or *parameters*) should take on particular values each time the function runs. This gives us an enormous amount of expressive power, since the same steps repeated with different values can give us dramatically different results.

- Functions Part 1 *(9m45s)*
- Functions Part 2 *(11m55s)*
- Using Functions Part 1 *(7m29s)*
- Using Functions Part 2 *(9m53s)*
- Locals and Globals Part 1 *(10m11s)*
- Locals and Globals Part 2 *(10m27s)*
- Useful Globals *(7m39s)*

Group 4: If Statements. Every time you run an interactive program, you can get a different result. A central mechanism for giving your program this kind of flexibility is the *if statement*.

If Statement Overview *(3m12s)*

Booleans *(6m30s)*

If Statements Part 1 *(10m54s)*

If Statements Part 2 *(9m52s)*

Logical Tests Part 1 *(7m41s)*

Logical Tests Part 2 *(6m6s)*

Using Logical Tests Part 1 *(12m11s)*

Using Logical Tests Part 2 *(10m54s)*

Combining If Statements Part 1 *(8m26s)*

Combining If Statements Part 2 *(11m34s)*

Group 5: The Keyboard. Here we see how to write code that responds when the user has pressed a key on the keyboard, and take different actions depending on which key it was. We also see how to respond to when a key is released, and how to handle a few special but common situations.

char *(4m10s)*

Keyboard Part 1 *(10m37s)*

Keyboard Part 2 *(11m33s)*

Keyboard AutoRepeat *(9m4s)*

Keyboard Extras *(7m14s)*

Group 6: Examples. This week we've covered a lot of material. Here we put the pieces together in two examples.

A Bouncing Ball Part 1 *(9m46s)*

A Bouncing Ball Part 2 *(10m39s)*

A Bouncing Ball Part 3 *(4m23s)*

A Little Game Part 1 *(9m52s)*

A Little Game Part 2 *(10m41s)*

Group 7: Homework and Recap. We survey what we've seen this week, and then you get to put it into action in your homework.

Week 3 Recap *(4m48s)*

Week 3 Homework *(7m8s)*

WEEK 4: CONTROL

This week is all about control: gaining more control over the graphics we draw, controlling how our programs run, and controlling our programs with the mouse. We'll also look at a concept that lets us think about how to structure animation, and look at some of Processing's built-in functions that can make our life more convenient.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 4 Overview *(8m38s)*

Group 2: Graphics Primitives. We've come a long way with lines, boxes, and circles, but there are many more kinds of shapes available to draw with. We'll also look at the idea of an "arrow", which isn't a built-in shape, but a powerful conceptual tool for creating shapes.

Points *(3m20s)*

Triangles *(3m50s)*

Quads *(4m56s)*

Arcs *(12m40s)*

Angles Part 1 *(10m5s)*

Angles Part 2 *(6m42s)*

Degrees and Radians Part 1 *(6m31s)*

Degrees and Radians Part 2 *(7m14s)*

Shapes *(9m27s)*

Arrows *(10m53s)*

Group 3: If Statements Revisited. The if statement is so handy, there are shortcuts available for two special types of situations that crop up frequently.

If Statements Revisited *(13m1s)*

Group 4: Loops. A loop allows us to repeat a bunch of lines of our program over and over, while changing one or more variables each time. This is something like calling a function over and over with different arguments, but it's far more convenient when we want to repeat it many times. We'll look at two different kinds of loops: the while loop and the for loop. Each type of loop can do everything the other type can do; the two varieties offer us two different ways to think about how we get the same work done.

While Loops Part 1 *(9m1s)*

While Loops Part 2 *(10m7s)*

While Loops Revisited *(12m56s)*

While Loops Example *(10m4s)*

For Loops *(10m40s)*

For Loops Revisited *(10m1s)*

Break and Continue *(2m18s)*

Break (*10m27s)*

Continue *(11m20s)*

Group 5: The Mouse. Touch screens are becoming more popular every day, but the mouse is still an important input device. And many touch-screen systems report touch information in a way very similar to the way mouse information is reported. Here we look at how to respond when the user moves the mouse, or pushes or releases a mouse button.

The Mouse *(12m17s)*

Using The Mouse Part 1 *(10m3s)*

Using The Mouse Part 2 *(7m37s)*

Dragging *(5m8s)*

Group 6: State. When we create animation, we're only drawing one frame at a time. In order to draw the correct image for each frame, we need to remember where we are in every aspect of the animation. We use the term state to refer collectively to all of this remembered information.

Animating With State *(10m37s)*

Using State *(11m11s)*

Group 7: Useful Functions. Processing offers us a variety of built-in functions that can save us a lot of time and effort. We survey them here and see some applications of them.

Map *(10m50s)*

Using map *(9m30s)*

Lerp *(12m16s)*

lerpColor *(7m52s)*

Dist *(11m30s)*

Useful Functions *(10m59s)*

Group 8: Recap and Homework. We survey what we've seen this week, and then you get to put it into action in your homework.

Week 4 Recap and Homework *(7m3s)*

Group 9: Supplements. The videos in this section are optional. They're here to illuminate interesting ideas, answer some questions you might have wondered about, and generally flesh out some of the topics we've seen.

Stroke Joins *(5m53s)*

Modes *(7m59s)*

Writing Fox And Hen Part 1 *(9m34s)*

Writing Fox And Hen Part 2 *(9m33s)*

Writing Fox And Hen Part 3 *(5m1s)*

WEEK 5: LISTS AND TRANSFORMS

We'll start out this week by looking at *random numbers*. By asking the computer to give us an unpredictable number from within a given range, we can make each run of our program different from every other. With random numbers under our belts, we'll then turn to our two main topics: lists and transforms.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 5 Overview *(7m45s)*

Group 2: Randomness. Random numbers are a great way to add some surprise and unpredictability to our programs. We can ask the computer to provide us with a number from within a given range, and then we can use that number in any way we want, from setting the location or shape of an object to controlling how it moves or behaves.

Random Numbers *(10m5s)*

Random Seeds *(10m53s)*

Flower Garden *(12m10s)*

Group 3: Arrays. The programmer's name for a list is an *array*. In Processing, you can create a list, or an array, of any type of data: ints, floats, colors, and so on, where all the elements of an array are of the same type. Arrays are a flexible and powerful way to retain information about a lot of similar objects at one time, and manipulate them all in similar ways.

Arrays *(5m33s)*

Creating Arrays Part 1 *(11m39s)*

Creating Arrays Part 2 *(10m45s)*

Creating Arrays Part 3 *(5m59s)*

Using Multiple Arrays Part 1 *(9m10s)*

Using Multiple Arrays Part 2 *(9m45s)*

Copying Arrays *(11m13s)*

Array Operations Part 1 *(9m3s)*

Array Operations Part 2 *(6m4s)*

Modulo Part 1 *(9m54s)*

Modulo Part 2 *(5m39s)*

Arrays and Modulo *(8m28s)*

Arrays and Random Numbers *(9m2s)*

Selecting With Arrays *(10m52s)*

Group 4: Transforms. We can manipulate the coordinate system by using *transforms*, or *transformations*. When we move, scale, or rotate the coordinate system, then everything we draw from then on will be affected by that transformation (everything that's already been drawn is unaffected). This is a simple but subtle idea that gives us a tremendous amount of power to draw rich imagery, but like any powerful tool, we have to use it carefully.

Transforms Overview *(10m32s)*

Translate *(10m17s)*

Rotate *(10m38s)*

Scale *(10m49s)*

Scale and Stroke *(8m16s)*

Group 5: Using Transforms. Transforms are powerful, but they can be tricky. We'll discuss how certain types of scaling and rotation can introduce *skew*, which can make objects appear to be leaning. We'll also look at the *transformation stack*, which provides the key technique for using transformations efficiently.

Transform Interactions *(12m4s)*

Transforms In Action *(10m58s)*

Stacks *(6m34s)*

The Transform Stack Part 1 *(9m41s)*

The Transform Stack Part 2 *(6m6s)*

The Transform Stack In Action Part 1 *(8m16s)*

The Transform Stack In Action Part 2 *(6m33s)*

Group 6: Recap and Homework. We survey what we've seen this week, and then you get to put it into action in your homework.

Week 5 Recap and Homework *(8m52s)*

WEEK 6: CURVES

Week 6 is all about creating, drawing, and using free-form curves. So far, the only curves we've been able have been circles and ellipses, and pieces of them. Now we extend our range to create big, smooth curves that can take on of any shape we want. Yet they're always under our control, and we can adjust their shape with as much precision as we desire.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 6 Overview *(4m27s)*

Group 2: Curves and Segments. An idea that's common to both types of curves is that we draw them out of *segments*. A segment is a single little bit of a curve that's defined by just four points. The curve only goes through two of these points. The other two serve as controls to influence the shape of the curve.

Curves and Segments *(6m54s)*

Group 3: Useful Geometry Tools. When we work with curves we'll frequently generate the points that control their shape by using combinations of other points. We'll look at a few nice tools, ideas, and shortcuts that can make this everyday job much easier.

Two Geometry Tools *(3m42s)*

Scaling Lines *(6m8s)*

Rotating A Line 90 Degrees *(6m46s)*

Normalizing A Line *(7m57s)*

Using the Geometry *(8m13s)*

Group 4: Catmull-Rom Curves. We look at *Catmull-Rom* (or CR) curves (named for the two men who developed the underlying mathematics). A CR curve segment contains four points: the first and last points influence the shape of the curve, which is drawn between the second and third points. We can draw big, complicated, smooth curves by drawing lots of segments. We can also find out lots of useful information about a curve. One of the most important things we can learn is the location of points along the curve, which lets us use the curve to control the on-screen motion of other objects.

CR Curves Intro *(10m24s)*

Closed CR Curves *(11m24s)*

CR Curve Segments *(11m1s)*

CR Curve Utilities *(10m56s)*

CR Offset Curves *(12m14s)*

Group 5: Bézier Curves. We turn now to *Bézier* curves (named for the man who popularized the underlying mathematics). A Bézier curve segment contains four points: the curve is drawn between the first and last points, and the middle two points serve as "handles" that influence the shape of the drawn curve. Using the handles we can produce a wide variety of curves between the first and last points. We can draw big, complicated, smooth curves by drawing lots of segments. We can find points along the curve, and find the tangents and normals that let us build offset curves.

Bezier Curves Intro *(8m40s)*

Bezier Curve Smoothness *(9m13s)*

Bezier Big Curves *(11m22s)*

Bezier Curve Segments *(2m12s)*

Bezier Curve Points Part 1 *(7m51s)*

Bezier Curve Points Part 2 *(5m30s)*

Bezier Curve Motion *(9m40s)*

Bezier Offset Curves *(10m29s)*

Group 6: When you want to rotate something to point at something else, you need to find the angle by which to rotate it. The easiest way to do this is almost always to call the built-in function `atan2()`. The `atan2()` function is useful for both types of curves.

Rotating with atan2 *(9m39s)*

Using atan2 *(10m0s)*

Group 7: Recap and Homework. We survey what we've seen this week, and then you get to put it into action in your homework.

Curve Examples *(4m8s)*

Curves Recap *(11m36s)*

Week 6 Homework *(6m47s)*

WEEK 7: TYPE, IMAGES, AND PATTERNS

This week we introduce two new graphics objects, and new tools to help us create animation. We first dig into Processing's system for drawing and manipulating type on the screen. Then we see how to read in and manipulate images (like photographs) in image formats like jpg and tiff. We'll see how to read and write individual pixels on the screen to change our pictures. Then we'll look at ways to create repeating patterns of numbers. We'll also encounter two new data types: one for holding strings of characters, and one for conveniently representing both the x and y values of a point in a single variable.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 7 Overview *(5m17s)*

Group 2: Strings. The *String* is a built-in data type for storing character strings. It's much better than making your own array of characters, because it's supported by a bunch of useful, built-in operations.

Strings Part 1 *(8m2s)*

Strings Part 2 *(8m20s)*

String Tools *(6m34s)*

Group 3: Typography. Placing type on the screen is great for lots of different kinds of sketches. Processing has its own way of dealing with typefaces and setting type on the screen.

Typography Essentials *(11m22s)*

Typesetting *(9m54s)*

Measuring Type *(8m12s)*

Typography In Action *(5m14s)*

Why Font Files *(5m22s)*

Group 4: Images. It's great to be able to show photographs and other saved images in our sketches. We see that it's easy to load an image file into our sketch and display it. We then see how to individually read and write pixels of the on-screen image.

Images *(7m6s)*

Get and Set *(7m55s)*

The pixels Array *(11m17s)*

Comparing Pixel Methods *(10m34s)*

Group 5: PVector. When working with screen points and locations, we need to maintain both the x and y components of those points. Processing offers a nice, built-in data type called the *PVector* that lets us bundle up both values into a single variable.

PVector *(7m11s)*

PVector Arrays Part 1 *(5m56s)*

PVector Arrays Part 2 *(9m56s)*

PVector Tools *(8m7s)*

Group 6: Repeating Patterns. Patterns of numbers are an important building block for creating animation. We can use some patterns immediately for simple animation, and we can combine patterns to make more complicated sequences for more subtle or interesting animation. You don't have to be a numbers whiz to use these patterns; it's enough to get a feeling for them and then just explore, trying out different ways to mix them together.

Creating Repeating Patterns *(10m32s)*

Five Repeating Patterns Part 1 *(7m39s)*

Five Repeating Patterns Part 2 *(5m49s)*

Group 7: Recap and Homework. We look over what we've seen this week, and then you get to put it into action!

Week 7 Recap *(10m49s)*

Week 7 Homework *(2m1s)*

WEEK 8: OBJECTS

This week we'll start by reviewing how to break up a big program into several files. Then we'll dig into our big topic: *object-oriented programming*. This is a very powerful way to think about and write programs that contain lots of similar, and potentially interacting, objects. Then we'll look at a few things that we haven't discussed yet: how to draw graphics into offscreen memory (so you can easily re-use them), finding the time and date, reading and writing text files, and a brief look at Processing's 3D abilities. We'll wrap up by looking at how you can share your work with other people, and look at to some ways you can expand your range and explore new kinds of applications.

Group 1: Overview. We start by looking at this week's context, so you have an overview of where we're going and how the pieces all fit together.

Week 8 Overview *(9m8s)*

Group 2: Big Projects. When we write a big project, it's often handy to break it up into multiple source files. This technique will be particularly useful when we write our own objects.

Big Projects *(6m38s)*

Group 3: Object-Oriented Programming. The principles of *object-oriented programming* let us organize and write our programs in a new and powerful way. There's nothing you can do with these techniques that you can't already do, but they offer a huge conceptual advantage when your programs get big.

Objects Overview Part 1 *(6m52s)*

Objects Overview Part 2 *(8m46s)*

Creating Objects *(9m23s)*

Implementing Objects *(8m7s)*

Encapsulation *(10m57s)*

Arrays of Objects *(12m25s)*

Subclasses Part 1 *(9m51s)*

Subclasses Part 2 *(8m14s)*

Drawing With Subclasses *(8m41s)*

Subclasses and Variables *(8m35s)*

Subclass Example Part 1 *(10m50s)*

Subclass Example Part 2 *(12m51s)*

Group 4: Odds and Ends. In this section we cover three topics that didn't have a natural fit in the previous weeks. Offscreen drawing lets us draw images once and re-use them, potentially saving us a lot of time, and thus speeding up our programs. We'll also see how to get the current time and date, and how to read and write text files.

Offscreen Drawing Part 1 *(9m52s)*

Offscreen Drawing Part 2 *(8m50s)*

Time and Date *(10m48s)*

Disk Files Part 1 (*12m0s)*

Disk Files Part 2 (*10m5s)*

Group 5: Overview of 3D. Processing offers some basic tools for creating and displaying shapes in 3D. This subject is very big, and we consider it an advanced technique. This video, and the examples, give you a taste of what you can do, and some starting points if you want to dig into it yourself.

Overview of 3D *(6m35s)*

Group 6: Sharing Your Work. We cover some of the popular ways to share your Processing programs with other people.

Sharing Your Sketch *(10m46s)*

Group 7: Going Further. We've covered a huge amount of information in this course! If you're thirsty for more, here are some ideas for your next.

Going Further *(8m17s)*

Group 8: Recap and Homework. We review this week's material, and see the new assignment.

Week 8 Recap *(9m25s)*

Week 8 Homework *(2m44s)*