



Unit 1

Videogames and Coordinate Planes

Unit Overview

Students discuss the components of their favorite videogames, and discover that they can be reduced to a series of coordinates. They then explore coordinates in Cartesian space, and identify the coordinates for the characters in a game at various points in time. Once they are comfortable with coordinates, they brainstorm their own games and create sample coordinate lists for different points in time in their own game.

Learning Objectives

Students will:

- Learn to work as a team
- Learn the basic expectations and goals of the class
- Learn the Cartesian coordinate system

Product Outcomes

- Students - in teams - will successfully reverse-engineer a simple side-scrolling game.
- Teams will brainstorm their own games and generate the geometry for their games at different points in time.

State Standards

See [Bootstrap Standards Matrix](#) provided as part of the Bootstrap curriculum.

Length: 90min

Materials and Equipment

- Student [workbook](#) folders - in pairs! - with names on covers.
- Computer, connected to a projector, with "NinjaCat" preloaded [[DrRacket](#) | [WeScheme](#)].
- Pens/pencils for the students, fresh whiteboard markers for teachers
- Cutouts of [NinjaCat](#) and the [Ruby](#)
- Class posters (List of rules, language table, course calendar)
- Language Table (see below)

Preparation

- Distribute workbooks, pens, arrange chairs so students are in pairs
- Set up student machines and projector, and check to make sure it all works!
- Post the agenda, rules, calendar, basic skills, language table, etc

Language Table (empty)

Types	Functions

Agenda

- 5min [Intro & Expectations](#)
- 10min [Dissecting a Demo](#)
- 15min [Coordinate Planes](#)
- 10min [More dissection](#)
- 15min [Brainstorming](#)
- 15min [Intro to Racket](#)
- 20min [Circles of Evaluation](#)
- 5min [Closing](#)

Introduction and Expectations**Time: 5 minutes**

- Welcome to your first day of Bootstrap!
- *Introduce the teaching staff. Give some background: age, where you're from, something surprising about yourself, favorite food, etc. Anything to allow kids to connect. Ask kids for their names!*
- I want to set some expectations here, to make sure we're all on the same page.
- In this class, you're going to be working on some very difficult material - you'll be using a language that is taught to college kids, and they're a lot older than you.
- You're going to need to be very focused, and disciplined about solving the problems we'll be throwing at you.
- Because this is the same material that they teach in college, we're going to expect you to be able to act like college students. What do you think that means?
- *Take 2-3 suggestions.*
- Okay, so here are the rules: (point to a visual while going over this)
 - Unless we say otherwise, raise your hand to speak
 - The computers are for programming, not for Facebook, chatting, email or playing around
 - If you're not programming, the computer monitors need to be OFF. When we say "monitors off," everyone should shut off the monitors, stop talking, and give their attention to the teachers. Let's try this one out - everyone make sure the monitors are on ... now start talking ... louder! ... MONITORS OFF! Repeat as necessary.
 - No one in this class is allowed to put each other down. I don't ever want to hear anyone call somebody else "stupid," "idiot", or anything else. Don't tell each other to shut up, and be respectful to everyone. Is that clear?
- When it comes time to program your games, you'll be working in pairs - you and a friend. But during class, there are a lot of activities and games we'll be playing where you and your partner will be part of a group.
- Right now, all of you are sitting with your group, clustered together around the classroom. When it comes time for team competitions or group activities, you'll be working with the people sitting nearby.
- These are just your groups for today! Depending on how well you work together, we'll be able to make some adjustments from class to class.
- *Assign group names (tip: use names that have something to do with computer science, or college, etc)*

Dissecting a Demo**Time: 10 minutes**

- Soon you're going to be designing your own games, but it's important to know how a simple game is built before you start designing your own.
- Let's take a look at a real game, which I've made using the same language and tools that you guys will be learning. When you're done with this class, you'll be able to make something similar.
- *Show the kids "NinjaCat". Be sure to end on a frozen frame of the game, so the whole class can see the same image.*
- Turn to **page 1** in your workbooks. Here we have a table, which we'll use to reverse-engineer, NinjaCat and see how it works. Raise your hand if you can tell me one thing that you saw in the game. Wait until a few hands are up, then call one. Excellent! So we can write that down in our first column.
- In your groups, take one minute to come up with a complete list of all things in the game. Your group will get a point for each thing they can find. Everyone in your group should have this list written down - not just one person! If even one person in your group hasn't written it down, the group doesn't get the point! GO!
- *During the minute, walk around and see how groups are doing. This is the time to encourage expectations of community and respect - call out good teamwork when you see it! When time is up, give them a countdown: "30...10... 5... 4... 3... 2... 1... PENCILS DOWN, EYES UP HERE!" Wait for total silence and complete attention.*
- *Have groups volunteer some of their answers and write them on the board. When they start listing items in the background, explain that you'll be grouping them all together into "background." Assign scores to the board: "team MIT is tied with team Northeastern!"*
- Okay, so we've got our list. Now we need to think about what is changing when we play the game. What about the Ruby? Does it get bigger? Does it change color? Does it spin around?
- The only thing that changes about the Ruby is its position! Everything else about it is the same.
- In your groups, take one minute to fill in the second column, for each thing in your game. GO!



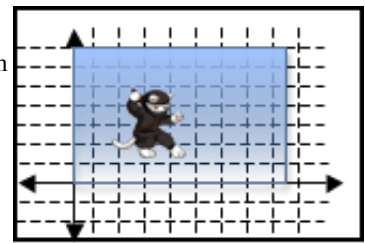
Thing	Changes
Cat	position
Clouds	position
Ruby	position
Dog	position
Background	nothing
Score	value

- Again, give a countdown: "30...10... 5... 4... 3... 2... 1... **PENCILS DOWN, EYES UP HERE!**" Wait for total silence, and complete attention.
- Quickly fill in the second column on the board. Assign scores.

Coordinate Planes

Time: 15 minutes

- What is a position? Suppose you were telling your friend about this picture on the phone. How would you tell her exactly where the Dog is? (Have students discuss.)
- Just like your friend on the phone, computers need to know exactly where something is in order to draw it on the screen. To do this, they use a coordinate system.
- (Draw a number line, with 10 tick marks).
- You've all seen number lines before, right? There are a bunch of evenly spaced markings on the line, and each one represents a number. A number line is actually a really simple coordinate system, which lets you find points in one dimension! For example, we can take our little cutout of NinjaCat, stick him anywhere on the line, and now we can all agree that he's sitting at 5, 6, or any other number on our line. Number lines can also have negative numbers, so we can put NinjaCat at -2, -8, or even 0.
- Our game has two dimensions, and we'll need a number line for both (draw a second number line). Let's call our first line, which runs from left to right the x-axis, and the new line, which runs up and down the y-axis. Now we have a grid, and we can stick NinjaCat anywhere on the grid. Let's do a quick example:
 - stick NinjaCat at (4, 6)
 - What is NinjaCat's position on the x-axis? To find out, we just drop a line down from where NinjaCat is, and read the position on the number line. He's at 4.
 - What is his position on the y-axis?
 - If time allows, invite volunteers up to the board to try placing NinjaCat and the Ruby at different locations.
- When we write down these coordinates, we always put the x before the y (just like in the alphabet!). Most of the time, you'll see coordinates written like this: (200, 50) meaning that the x-coordinate is 200 and the y-coordinate is 50.
- On the computer, the screen is all the parts of the grid from 0 to 640 on the x-axis, and 0 to 480 on the y-axis. Superimpose the screen over your coordinate plane.
- What is the coordinate for the lower left-hand corner of the screen? What about the lower right-hand corner? The center?
- Can you think of a coordinate that puts NinjaCat on the left-hand side of the screen? The top?
- Can you think of a coordinate that puts him off the screen?



More dissection

Time: 10 minutes

- So let's look back at our table (**page 1**). Now that we know what a position is, let's be more specific about what is changing during the game.
- Can NinjaCat move up and down in the game? Can he move left and right? So what's changing: his x-coordinate, his y-coordinate, or both?
- What about the clouds? Do they move up and down? Left and right? Fill in the rest of the table with your team.
- Terrific! Great job, all of you. Now, turn to **page 2** in your game planning workbooks and look at the project sheet that has a picture of the NinjaCat game you just saw.
- Raise your hand if you can tell me what the coordinates are of the upper-left-hand corner. Take volunteer. What about the bottom-right-hand corner? What about the coordinates of the point right in the center of the screen?
- In your groups, take a minute to find and label all of the midpoints - the points in the middle of each side of the screen: top, bottom, left and right. (You may want to point to these points on the board). Once again, your group only gets a point if everyone writes and labels these coordinates correctly. GO!
- Give a countdown, demand attention, then take volunteers. Assign scores, discuss.
- In the next round, your groups will have five minutes to fill in the coordinates for all the characters at the bottom of the page.
- Give a countdown, demand attention, then take volunteers. Assign scores, discuss.



Brainstorming!

Time: 15 minutes

- Okay, teams - great job. Now it's time to just work as pairs, with you and your partner.

- It's your chance to get creative!
- Just like we made a list of everything in the NinjaCat game, we're going to start with a list of everything in your games.
- To start, your game will have with four things in it:
 1. A Background, such as a forest, a city, space, etc.
 2. A Player, who can move when the user hits a key.
 3. A Target, which flies from the right to the left, and gives the player points for hitting it.
 4. A Danger, which flies from the right to the left, which the player must avoid.
- Suppose this was a sports game, like football. What might the player be? (A football player.) How about the goal? (A football.) The Danger? (Player from a rival team.)
- Maybe your game is about a puppy out for a walk. If the player is the puppy, what gives him points? What should he stay away from?
- On **page 3**, you'll find a planning template for you to make your own game.
- Put your **names** at the top of your proposal.
- Each pair needs to pick a setting for their game, and briefly sketch their characters. You don't need to make it perfect, but these drawings should be good enough to give other designers at your company a good idea of what you had in mind. You will have 5 minutes. GO!
- *Walk around and make sure to consult with every team. A lot of students will have trouble fitting their ideas into this format, or they'll struggle with coordinates. Be clear about what can and cannot be done! (e.g. - no 3d games, joysticks, multiplayer games, etc)*

Introduction to Racket and Numbers

Time: 15 minutes

- *Draw a big Circle on the board.*
- This circle is going to be our computer! We need to program it to do things, but once we do, we can use it to try out all of our programs.
- Look at our language table: right now we can't put anything in our circle, because our language has nothing in it. Let's add some values.
- On three, Let there be numbers! "1... 2... 3... Let there be numbers!"
- *Add "numbers" to the Types section of the language table.*
- Now we have numbers, which we can put in our Circle of Evaluation. (*Put in the number 4.*) What is the value of everything in this circle? Are you sure? (*Try other numbers, including decimals and negative numbers.*)
- Let's ask a real computer...
- *Have students open the editor on their computers. Refer to overhead projector as necessary.*
- This is a tool that allows you to write Racket programs. On your screens you'll notice two large boxes: the Definitions window at the top and the Interactions window at the bottom. For now, we're going to just focus on the Interactions window.
- The Interactions window is like scrap paper, where you can write short programs and try them out by hitting "Return." When you know what you want to keep as part of your finished product, you write that code up in the Definitions window. Look on the top of the window: you'll see a button with a green man running. What do you think happens when you click "Run"?
- Whenever you run a Racket program, it computes the program you've written - just like our circle.
- Type in the number 4, hit Run to see if the computer agrees with you. Congratulations: you just wrote your very first Racket program, and it came out to be the same value as what you'd expected! Try typing in other numbers and see what happens. What happens if you write a decimal?
- In English we have things like nouns and verbs. What's the difference between them? In Racket, anything that is a value is like a noun. I can give Racket the number 5 and it will give it right back, unchanged. All of the numbers you've entered are examples of values.
- Racket also has things that are like verbs, called Functions. I can throw a marker, the same way I can add two numbers.
- You already wrote programs that were just values. Now you're going to write programs that apply functions to those values.



```
Welcome to DrRacket, version 5.0 [3m].
Language: Beginning Student; memory limit: 256 MB.
> 4
4
```

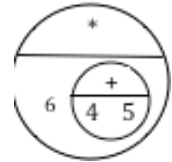
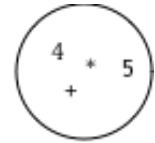
Circles of Evaluation

Time: 20 minutes

- *Write $4 + 5$ (jumbled, in no order in particular) in the Circle of Evaluation.*
- What is the value of the stuff in the circle? Let's figure out the math first. (*Write the equation.*) We know we're adding, so we start

with the plus sign. ("+" in the middle.) Then we add 5 and 4 on either side. ("5 + 4".) Does it matter if I write (4 + 5)?

- Let there be subtraction!
- Write a subtraction example in the Circle of Evaluation (again, jumbled), and ask students to evaluate it. Typically, students will realize that there are two possible answers! The lesson here is that the order of inputs matters. If necessary, have them type examples into the Interactions window to see that swapping the inputs gives different results!
- Write $4 + 5 * 6$ in a Circle of Evaluation, and ask students what they think the value is. As with subtraction, have them discover that order of operations matters, and use the actual formulas to demonstrate: (" $5*6 + 4$ ", " $4*5 + 6$ ")
- We need to improve our Circle of Evaluation, so we don't run into this problem. Let's use the two solutions we came up with:
 - 1) all circles have only function
 - 2) it matters in what order the arguments are written.
- We'll separate the function from the inputs by drawing a line between them: the function is above the line, the inputs are below.
- How do we compute this program? Well, our new rule tells us to look at the left first: what is the value of 6?
- Now we look at the right: that's a new circle, so we have to evaluate that first. What is the left-hand side? A 4! And the right? A 5! What are we doing to the 4 and 5? That's right, we're adding. What do we get when we add 4 and 5? (Replace rightmost circle with 9.)
- Now we have two numbers. What are we doing to 6 and 9? (Replace circle contents with 54.)
- We write this in Racket the same way. We always put parens before a function and after its inputs (Write the parens and the "*"). In order to apply this function, we need to compute the left side and then the right. The left side is the number 6, so we can just write that in. (`* 6 ____`)
- The right side happens to be another Circle of Evaluation (more parens), so we apply the same rules there: write the function and then look at the left and the right. These are numbers, so we can write them in directly. (`* 6 (+ 4 5)`)
- Try writing this in the Interactions window, and hit "enter". What did you end up with? Now try writing in the complex example we used.
- Did you get the same answer?
- With your partner, try to come up with more examples - can you figure out how to subtract three numbers?
- Let students discuss briefly, but usher them along to the solution using nested circles. They'll have plenty of practice soon!
- I can't put three numbers in the circle, because there's no way to tell the computer which two numbers should be subtracted first. Is it $(2-3)-5$, or $2-(3-5)$?
- Well, if a complete circle evaluates to a number, and numbers go inside circles, why not try putting circles inside circles?
- Have students practice circles - make it into a game! There are several exercises provided on **page 4** of the student workbook.



```
> (* (+ 4 5) 6)
54
```

Closing

Time: 5 minutes

- Who can tell us one thing we learned today?
- Who saw someone else in the class do something great?
- Cleanup, dismissal
- **NOTE TO INSTRUCTORS:**
 - Make sure student names are on page 3
 - Take page 3 itself, or take photos of page 3, to prep game images for **Unit 3**.
 - Images should be in PNG or GIF format. Background images should be 640x480, and character images should generally be no larger than 200px in either dimension. Make sure that the character images have transparent backgrounds!
 - TIP: use animated GIFs for the characters - not only does the animation make the game look a lot better, but these images usually have transparent backgrounds to begin with.
 - For more instructions on setting up the game files, read the [Coding Instructions](#) document.

Bootstrap by [Emmanuel Schanzer](#) is licensed under a [Creative Commons 3.0 Unported License](#). Based on a work at www.BootstrapWorld.org. Permissions beyond the scope



of this license may be available at schanzer@BootstrapWorld.org.