PROFESSOR: So you may recall that, in the last lecture, we more or less solved the drunken student problem, looking at a random walk. I now want to move on and discuss some variants of the random walk problem that are collectively known as biased random walks. So the notion here is, the walk is still stochastic but there is some bias in the direction, so the movements are not uniformly distributed or equally distributed in all directions. As I go through this, I want you to sort of, think in advance about what the take-home message should be. One, and this is probably the most important part for today, is I want to illustrate how by designing our programs in a nice way around classes, we can change them to do something else writing a minimal amount of code. And the idea here is that classes are not there just to provide some syntax that makes life complicated, but to actually provide a mechanism that lets us structure our programs in a way that we can modify them later. I also want to give you a bit more experience looking at data, and understanding how, when you get the results of a simulation or anything else that provides you with data, you can plot, them study the plots, and try and use those to develop an understanding about what's going on. And finally, I want to just get you thinking more about random behavior. Because we're going to talk a lot more this semester and in general throughout your careers, you'll see a lot of uses of stochastic things.

All right, so let's think about these biased walks. So for example, assume that the drunk grew up in South Florida, and really hates the New England winter. Well there might be a bias that even though the drunk is not in total control, she's kind of wandering southward. So there's a bias when taking a step to more likely go south than certainly north. Or imagine that the drunk is photosensitive and moves either towards the sun or away from the sun, or some such thing. So we're going to look at different things of that nature and think about, I'll come back to Chutes and Ladders later in the lecture, you can speculate on why this is here. And we'll look at that. So, this is on the front page of your handout here.

I've taken the random walk program we looked at last time and changed it a bit. And in particular, I've changed the way the drunk works. So let's look at the code. Wow, haven't even done anything it's -- see how this works. So now, drunk itself is a very small class, one I don't actually intend to ever instantiate. I've pared it down, it has only two methods in it. It has an init, which is the same as before, and a move which is the same as before, almost the same. Now I'm going to have the class usual drunk. That's the drunk we looked at the last time. And this usual drunk will behave just like the drunk we looked at last time. In here, I'm going to override the move method of the inherited superclass, and what this is going to do is, it's going to make the same random choice it made before

and then it's going to call drunk dot move with the new compass point. So all I've done is, I've taken a couple of lines of code out of the previous implementation and moved it to the subclass. So it's now, excuse me, in the subclass I'm making the decision about which direction to move. Once I've made that decision, then I can call the drunk in the superclass. Notice here, when I've made the call, drunk dot move. So usually you're expecting me to write something like d dot move. The object. Here, instead I'm specifying which class to get the move from, So this is saying, don't call my local move, call the move from the superclass. There are a number of other syntaxes I could've used to accomplish the same thing. I used this because it seemed the most straightforward.

Now let's compare the usual drunk to the cold drunk. So the cold drunk, again, I've overridden the move, and now it does something different. It gets the random compass point as before, but if it happens to be south, it calls drunk dot move with twice the distance. So whenever I get the notion I'm moving south, because I love the warmth, I actually move twice as fast. So instead of taking one step, which I do to the other compass points, if I happen to be going south I'll take two steps. So I'm biased towards moving southward. And then I call drunk dot move. Otherwise, I call drunk dot move with the distances before. So I've isolated into the subclass this notion of being heat-seeking. We've got a third sub class, ew drunk, known for east west drunk. And here, what I've done is, I choose to compass point and then while it's not either east or west, I choose again. So this drunk can only move eastward or westward. Can't move north or south. So again you'll see I'm able to have three kinds of drunks. Who as we will see, exhibit rather different behavior from each other with a minimal amount of change to the code.

If we look down at the bottom, we'll see that the code is pretty much exactly what it was last time. By the way, this doesn't quite match the code in your handout. I decided to make the plots a little bit prettier with better titles and things, but when I tried to put these in your handouts it wouldn't fit. So I have a little more compact things, but this is, I think, a better way to write it. But essentially, perform trial, perform sim, answer quest, are all identical to what they were before. So because I structured my program in such a way that the drunk's behavior was independent of all the code for the generating the trials and plotting the results, I didn't have to change any of that. And this is the key to good design. Is to isolate decisions in small parts of your program, so when as inevitable happens, you want to enhance the program, or you discover you've done something wrong, the number changes you make is not large. OK, one thing we do need to look at down here is, it's not quite the same as before, because we'll see that there's an extra parameter to these things called drunk type. So if we start with answer question, it takes as before, the time, the number of trials, a title, and this one extra thing called the drunk type. And then when it goes on and it calls perform sim, it passes the drunk type. And this is the key line here. Before, you may recall we just said d equals drunk with the arguments. Here, I say d equals drunk type.

So what is drunk type? Drunk type is itself a type. We can use types, you know variables can have as their values

types, parameters can have as their values types. So I'm just passing the type around so that here, when it comes time to get a drunk, I get a drunk of the appropriate type. So I can now answer the question about a usual drunk, or a cold drunk, or a ew drunk. This is very common kind of programming paradigm. We've talked about it before, but this is yet another example of how we use polymorphism. Polymorphism. We write our code in such a way that it works with many types of objects. Here we're writing code that will work with any subtype of drunk. Now I'd better not pass it a float. You know, if I answer quest, and instead of something like usual drunk or cold drunk or e w drunk, I pass it float or int, the world will come crashing down around my head. But I haven't, so you'll see down here, I pass it num steps, number trials, and usual drunk. Which is the name of? A class. Any questions about this? Yes?

STUDENT: [INAUDIBLE]

PROFESSOR: Louder, please?

STUDENT: [INAUDIBLE]

PROFESSOR: Good, good question. The question is, does a drunk type make a new drunk? Now, I have so much candy to choose from, I don't know what to throw you. Do you like Tootsie Rolls, Snickers, Tootsie Roll Pops, Dots, what's your choice? Snickers. OK, so what is it? Down here nothing is created when that call was made. It was if I had typed the word float or int. I'm merely saying, here is a type, and the parameter should now be bound to that type. Now any time you use the parameter, it's just as if you had written the string, in this case, usual drunk. So when I get up to here, think about, it will take, the variable drunk type, evaluate it, it will evaluate to a type usual drunk, say, in the first call. And then it will, just as if I had written usual drunk, invoke the init method of drunk, and create a new drunk, but this drunk will be of type usual drunk rather than a drunk. Does that make sense? All right, so this avoids my having to write a whole bunch of tests, I could have passed a string, and said if the string said usual drunk, then d equals something. But here instead of that, I'm actually passing the type itself. OK? So other than the fact that I have to pass the type around, this is exactly the same as before.

Now just for fun, we saw that last time when I asked you what did usual drunk do, there was some confusion. Not confusion, just a bad guess about how the drunk would wander. So, let's try it again. What do you think about the cold drunk? Will the cold drunk get further from the origin than the usual drunk? Not further from the origin? Who thinks the cold drunk will be further away the end of 500 steps? Who thinks the cold drunk will not be further away? And I'll bet we can all agr -- got that right, I think -- and I'll bet you'll all agree that, with higher probability, the cold drunk will be south of the origin than north of the origin. More interesting, how about the ew drunk? So this is a drunk that only goes east or west, east or west. Will this drunk, well, first of all, what's the expected distance? Do you think it should be close to zero or not close to zero? Who thinks it should be close to 0? Who

thinks it won't be close to 0? Well you guys have all fallen for the same trick you fell for last time. Fooled you twice. Be careful, one more and you're out. As we'll see. But it's interesting.

So let's run this program and see what happens. I can take a just a second or two and we should get a bunch of plots. All right, so here's the usual drunk. 100 trials and just as before, wandered away. Moderately smooth, but of course little ups and downs, maybe suggesting averaging over 100 trials is not quite enough. I felt a little better if it were smoother, but the trend is pretty clear here, I think we probably don't doubt it. Well, look at cold drunk. Like a shot heading south. So here we see, there's bias, that if you get south you go twice as far makes a huge difference. Instead of being roughly 20 steps away, it's roughly 120 steps away. So we see, and this is something to think about, a small bias can end up making a huge difference over time. This is why casinos do so well. They don't have to have a very much of a bias in their favor on each roll of the dice in order to accumulate a lot of money over time. It's why if you just weigh your dice a little bit, so that they're not fair, you'll do very very well. I'm not recommending that. Kind of interesting. The e w drunk is pretty much the same distance away as the usual drunk. This is why we run these simulations, is because we learn things. Now, I have to confess, it fooled me, too. The first time I ran this, I looked at this and said, wait a minute, that's not what I expected. So I scratched my head, and thought about, all right, why is it behaving this way? This happens me a lot in my own research, that I crunch some data, do some plots, look at it, get surprised, and on the basis of that I learned something about the underlying system I'm modeling. Because it provokes me to think in ways I had not thought before I saw the data. This is an important lesson. You don't ignore the data, you try and explain the data.

So when I thought about what happened, I realize that while I might have started at the origin, and after the first step I'm equally likely to be here or here. Once I've taken the first step, now my expected location is not zero, right? The first step, my expected location was zero. Because equally likely to be plus 1 or minus 1. But once I've taken a step and I happen to be here, my expected location, if I'm here, is plus 1, it's not 0. Because half the time I'll get back to 0, but half the time I'll get to 2. So once I have moved away from where I've started, the expectation of where I'll end up has changed. This is kind of a profound thing to think about, because when we look at it, this kind of behavior explains a lot of surprising results in life. Where, if you get a small run of random events that take you far from where you started, it's hard to get back. So if you think about the stock market, for example, if you happen to get unlucky and have a few days or weeks or months, as the case may be, where you get a run of bad days, it's really hard to get back if all the movements after that are random. Because you've established a new baseline. And as we'll see later, when you start doing things at random, it's likely you'll get a few runs in one direction or another, and then you've got a new baseline, and it's unlikely to get back to where you were originally. We'll see a lot of examples of this. But that's what's happening here, and that's why it looks the way it does. That makes sense to people?

Well, so I had this theory, as I often do when I have theories, I don't believe them, so I decided I better write some more code to check it out. So, what I did is, I took the previous simulation and now instead of looking at different kinds of drunks, I got more aggressive in thinking about my analysis of the data. So again, there's a lesson here. I decided I wanted to analyze my data in new ways. And the beauty of it was, I could do that without changing either any of the code about the drunk, or any they of the code about performing the simulation, really. All I had to do was do a different kind of analysis to a first approximation. So, let's look at what I've done here. What I've decided to do here is not only plot, and again, I think you'll see this code, not only plot how far from the origin the drunk is over time, but I'm going to actually plot two other things. I'm going to use a scatter plot, I think I use scatter in this one, I do, to show where the drunk is at each period of time. Actually, just all the places the drunk ends up in each trial. This will give me a visualized way just see, OK, in my, say, 100 trials, how many the drunk ended up near the origin, how many of them ended up really far away? And just, allow me to visualize what's going on.

And then I'm going to also use a histogram to sort of summarize some things. To get a look at the distribution of the data. We've been just looking at average results, but it's often the case that the average does not really tell the whole story. You want to ask, well, how many of the drunks ended up really far away? How many of the drunks ended up where they started? You can't tell that from the averages. So you use a distribution, which tells you information about how many people ended up where. So we again saw that, in a less pleasant context, maybe, in looking at the quizzes, remember I sent you some plots about the quizzes, and I didn't just say, here was the average. I showed you the distribution of grades. That allowed you to see, not only where you were relative to the average, but whether you were an outlier. Was your grade much better than the average? Or were there a whole bunch of people around where you are? So there's a lot of information there, that will help us understand what's really going on.

So we'll run this now. Again, there's nothing very interesting here about how I did it, I just used some of these Pylab dot hist, which gives me a histogram, Pylab dot scatter, which gives me a scatter plot. I'll take 500 steps, 400 trials. I've done some more trials here, so we get some smoother lines, better distributions. And then I'm going to look at the usual drunk and ew drunk only, just to save some time. At least I think I am. My computer runs faster when it's plugged in. All right, but we have our figures here, let's look at them. So here's figure one, the usual drunk. Well, all right, as we've seen this 1000 times, or at least six. So doing what it usually does, not quite 20. This is kind of interesting. This is the final locations of the 400 trials of the usual drunk.

So we see some information here, that we really couldn't get from the averages. That there are more of these points near the middle, but there are certainly plenty scattered around. And we get to see that, they're pretty much symmetric around where we started. This is what we would have hoped, right? If we had looked at it, and we had seen all the points up here, or most of them up here, we would have said, wait a minute, there's something wrong

with my code. This was supposed to not be a biased walk, but an unbiased walk. And if I'd seen a bias in this population, I should have gotten nervous that I had some problems with my implementation. My simulation. So in addition to helping me understand results, this gives me some confidence that I really am doing an unbiased walk, right? Some over here, some over here. Not exactly the same, but close enough that I feel pretty comfortable.

And if I look at the distribution, which probably I could've if I worked really hard, gotten from the scatter plot, but I really didn't want to, we'll see that most of the drunks actually do end up pretty close to the origin. But that there are a few that are pretty far away. And again, it's close to symmetric around zero. Which gives me a good feeling that things are working the way they should. This, by the way, is what's called a normal distribution. Because once upon a time, people believed this was the way things usually happened. This is called either normal or Gaussian. The mathematician Gauss was one of the first people to write about this distribution. We'll come back to this later, and among other things, we'll try and see whether normal is really normal. But you see it a lot of times. And basically what it says is that most of the values hang out around the average, the middle. And there are a few outliers, and as we get further and further out, fewer and fewer points occur. So you'll see these kind of curves, of distributions, occurring a lot. Again, we'll come back to distributions in a week, or actually in a lecture or two. See a little bit more today, and then more later on.

All right, here's our east west drunk, again as before, drifting off to around 18. Well, look at these dots. Well, this makes me feel, that at least I know that the drunk is only moving east and west. And again you'll see, pretty dense in the middle and a few outliers. Managed to get a little further east than west, but that happens. East is always more attractive. What we might have guessed. And if we look at the distribution, again we can look at it, and I'm just giving you the east west values here, and again it's about the same. So, what do I want you to take away from these graphs? Different plots show you different things. So I'm not going to try and claim that of these three different plots we did, one is the right one to do. What I'm going to assert is that you want to do all of them. Try and visualize the data in multiple ways. And sometimes it's much easier to, even if different plots contain exactly the same information, and in some sense of the scatter plot and the histogram do contain exactly the same information. In fact, there's more information in the scatter plot, in some sense, because I see each one individually. But it's, for me, a lot harder to see. If I'd ask you from the scatter plot, are they normally distributed? You might have had a hard time answering it. Particularly for the usual drunk, where you just saw a lot of points. So the idea of some summarizing it in the histogram makes it much easier to make sense of the data, and see what you're doing, and to try and learn the lesson you're trying to learn. OK, this makes sense to people?

All right, let's move on. Now we'll go back to this lovely picture we had before. Any of you ever play the game Chutes and Ladders? Raise your hand? OK, some things never go out of style. Well, imagine your poor drunk wandering through this board, and every once in a while, because I'm kind of a, not a nice person, I've eliminated

all the ladders, and have only some chutes. And every once in a while the drunk hits a chute and goes whoosh. And in fact, let's, for the sake of simplicity here, assume that the chutes are going to, every once in a while, the drunk will hit a bad spot, and go right back to the origin. So this poor, say, heat-seeking drunk who's trying to head south, does it for a while and maybe gets zipped right back to where he or she started. All right, I've now told you I want to put some, make this happen. What part of the code do you think I should change? Somebody?

STUDENT: The field.

PROFESSOR: The field, absolutely. Those are good to throw short distances. The field, right, because what we're talking about here is not a property of the drunk, or a property of the simulation, but a property of the space in which the drunk is wandering. So again, we see that by structuring the initial program around the natural abstractions that occur in the problem, natural kinds of changes can be localized. So let's go and change the field. So I'm going to have the field I had before plus this thing called an odd field. Odd field will be a subclass of field. This is not odd as in odd or even, this is odd as in strange. So I'm going to first define where my chutes are. So is chute will take, will get the coordinates, and assign it to x and y, so it gets the coordinates of a place, and then will return abs x minus abs y is equal to zero. So, where are my chutes going to occur? Somebody? Yeah, kind of radiating out from the origin. Any place, so I'll have my origin, here. Got my graph. And all along here, any place x and y are equal, there'll be a chute. Now, I have to be a little bit careful that it doesn't happen here, or we won't get anywhere.

So, and then move, in odd field, will call field dot move, which is unchanged from what it was before. And then it will say, if self dot is chute, set the location back to 0,0. So the poor drunk will move as before, and if he or she hits this wormhole, get instantly teleported back to the origin. Very small change, but it will give us rather different behaviors, at least I think it will. And then if we look at how we use it, not much changes. Except, what do you think is going to have to change? Where will I have to make the change, somebody? You can actually see the code here. I'll have to get a different field, right? So, at the place where I instantiate a field, instead of instantiating a field, I'll instantiate an odd field. Now if I'd had 16 different kinds of odd fields, or even two, I might well have done what I did with drunk. But in order to sort of minimize things, I've done something much smaller. So let's see, where did we get the field? I don't even remember. So, here's what I do when I'm looking at my program and I want to see where something is. I use the text editor, and I'm going to search for it. Let's see. Well, it's not there. There it is.

So here when I get my field, I get my drunk, and then I get a field which is an odd field. All right, anyone want to speculate what will happen when I run this one? Think the drunk will be closer or further from the origin when we're done? Who thinks closer? Who thinks further? Who thinks no difference? Well, you're right, I mean it's sort of logical if we think about it, that if every once in a while you get to zipped, back it's going to be harder to get

further away. We'll see different behaviors, by the way, it depends on the drunk, right? That's true for the usual drunk, but maybe not for the east west drunk. We can see what happens. So let's try some. We'll do it for the usual drunk, which is the more interesting case to start with. So you'll see whereas before we were up just shy of 20 most of the time, here we don't get very much much, further. We do get steadily further, but at a lot slower pace. What do you think it means that, before you remember, we saw a pretty smooth line. Here we see what may look you like a fat line, but is in fact a line with a lot of wiggles in it. What does that imply, about what's going on in the simulation, the fact that line is jagged rather than smooth as it was before? Pardon? Can't hear you?

STUDENT: That it jumps.

PROFESSOR: Well, certainly it happens because of the jumps, but what else is going on? But what is it, sort of, imply more, whoa, sorry about that, more generally? Right, what it's saying, is that because of the jumps, there's a lot more variation from trial to trial. Not surprising, right, because you sometimes you hit these wormholes and sometimes you don't. So let's look at this other figure. This is kind of interesting. So here's the scatter plot. And if we zoom in on it, what we see here is, there are essentially holes, you know, these alleys where no points lie. Just as we might have guessed from here. The only way you would get a point lying on these alleys, if it happened to be the very last step of the simulation, not even then, because the last thing we do is go back to the origin. So we can look at this, and say, well, sure enough, this is keeping people off of a certain part of the field. All right, some of you will be happy to know we are, for the moment, actually maybe for the whole rest of the term, leaving the notion of drunks. Though not of random walks. All right, any questions about what's going on here? And again, I would urge you to sort of study the code and the way it's been factored to accomplish these things. And play with it, and look at what you get from the different plots.

I now want to pull back from this specific example, and spend a few minutes talking about simulation in general. Computer simulation really grew hand in hand with the development of the computers, from the very beginning. The first large-scale deployment of computer simulation was as part of the Manhattan Project. This was done during the war to model the process of nuclear detonation. And, in fact, what they did was a simulation of 12 hard spheres, and what would happen when they would bump into each other. It was a huge step forward to do it with simulation, since the whole project had been stalled by their attempt to do this analytically. They were unable to actually solve the problem analytically. And it was only when they hit upon the notion of using a computer, a relatively new tool in those days, to simulate it, that they were able to get the answers they needed. And they did it using something called a Monte Carlo simulation. Now in fact, that's just what we've been doing with the random walk, the Monte Carlo simulation, and I'll come back to that in a minute. In general, the thing to think about, is we use simulation when we really can't get a closed form analytic solution. If you can put down a system of equations and easily solve it to get an answer, that's usually the right thing to do. But when we can't easily do that, the right

thing to do is typically to fall back on simulation. Sometimes even when we can do it analytically, simulation has some advantages, as we'll be seeing as we go forward. What we're typically doing when we're simulating anything, is we're attempting to generate a sample of representative scenarios. Because an exhaustive enumeration of all possible states would be impossible. So again, if sometimes you can't solve it analytically, you can exhaustively enumerate the space and then see what's going on. But again, usually you can't. And so we look for a sample, and the key is, it's gotta be representative. Of what we would get in reality. We'll see an example of that shortly. So simulation attempts to build an experimental device that will act like the real system in important aspects. So I always think of a simulation as an experimental device. And every time I run it, I'm running an experiment designed to give me some information about the real world. These things are enormously popular, so if you were to get on Google and look at simulation. So for, example, we could Google simulation finance. You know, we see we get about 3,000,000 hits. We can do biology. We get twice as many hits, showing the relative importance in the world of biology and finance. For all you Course 7 majors, we should make sure that should compare biology to physics. Oh dear, we'll see that physics is even more important than biology. And I know we have a lot of Course 2 students in here, so let's try mechanical. Oh, lot of those, too. And if I did baseball, or football, we'd get more than any of them. Showing the real importance of things in the world.

As we look at simulations, I want you to keep in mind that they are typically descriptive not prescriptive. So by that I mean, they describe a situation, they don't tell you what the answer is. So another way to think about this is, a simulation is not an optimization procedure. When we looked at optimization, that was prescriptive. We ran an optimization algorithm, and it gave us the best possible solution. A simulation typically doesn't give you this best possible solution, but if you give it the starting point, it will tell you the consequences of that. Now, can we use simulation to do optimization? Absolutely. For example, we can use it to do guess and check. Where we guess a, probably not to get a truly optimal solution, but to get a good solution, we can guess various possibilities, simulate them, and see. People do that all the time. If they want to see what's the right number of checkout counters at the supermarket, or what's the right airline schedule to use? They'll make a guess, they'll simulate it, and they'll say all right, this was better than this, so we'll choose it. All right, I'm going to stop here. Next time we're going to get more deeply into the actual stochastics, the probability distributions, and start understanding what's going on under the covers.