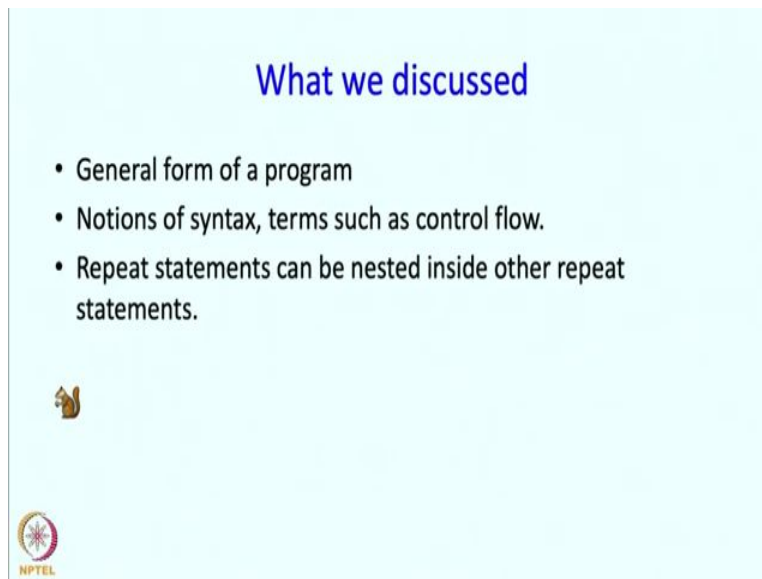



An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 1 Part – 4
Introduction
About the spirit of the course, demo


(Refer Slide Time: 0:27)



What we discussed

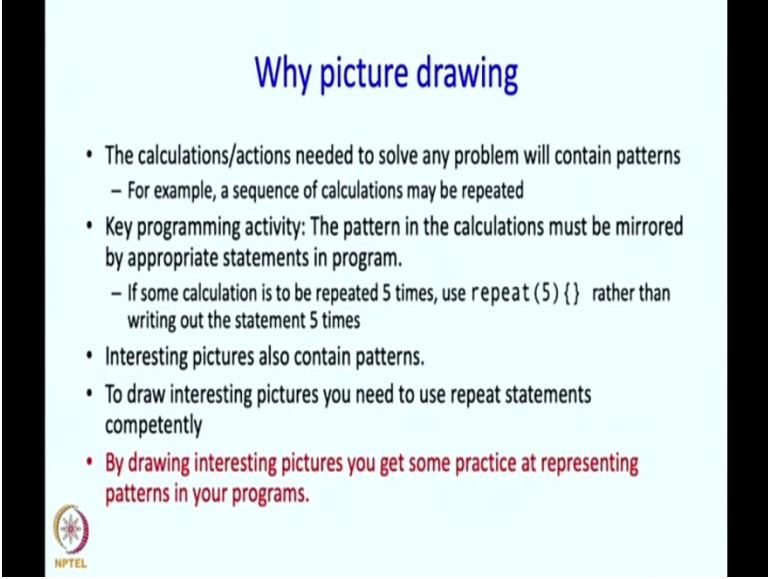
- General form of a program
- Notions of syntax, terms such as control flow.
- Repeat statements can be nested inside other repeat statements.






In the previous segment, we discussed the general form of a program, we discussed the notions of syntax and terms such as control flow, then we also observed that repeat statements can be nested inside other repeat statements.

(Refer Slide Time: 1:16)



Why picture drawing

- The calculations/actions needed to solve any problem will contain patterns
 - For example, a sequence of calculations may be repeated
- Key programming activity: The pattern in the calculations must be mirrored by appropriate statements in program.
 - If some calculation is to be repeated 5 times, use `repeat (5) { }` rather than writing out the statement 5 times
- Interesting pictures also contain patterns.
- To draw interesting pictures you need to use repeat statements competently
- **By drawing interesting pictures you get some practice at representing patterns in your programs.**



In this segment, we are going to take a review of what has happened, and we will answer some questions that you might have in your minds. So for example, we have been doing picture drawing in this lecture sequence and let me now say why we are doing picture drawing. So, let us ask what happens in general when you solve any problem. So when you solve any problem, you will need to do some calculations, you will need to take some actions.

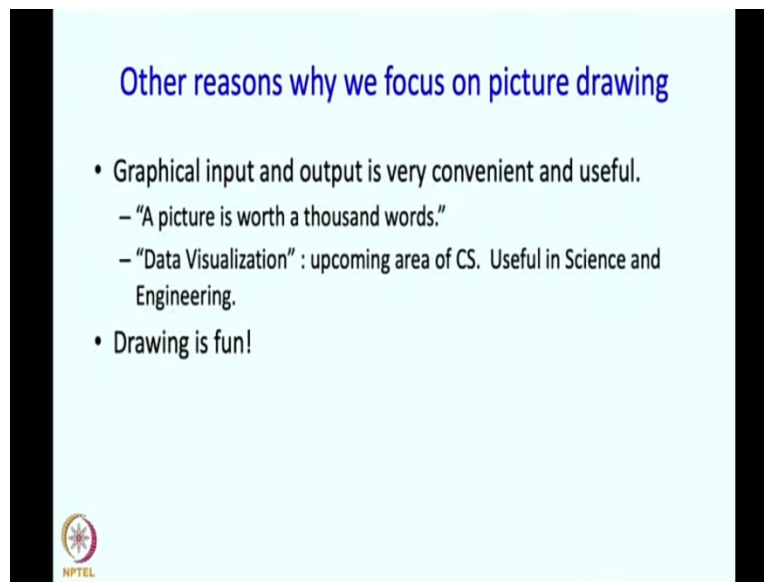
Now those actions will typically contain patterns, and you will have to understand those patterns. And a pattern might be something like a certain sequence of calculations is repeated. So what do you do in that case? In that case you use a repeat statement. So this is a key programming activity. The pattern in the calculation must be mirrored by an appropriate statement in the program.

So if it is a simple repetition, then you should have a repeat statement okay. If something is to be repeated 5 times you should not write that 5 times, but you should write a repeat statement with repeat count 5 and put whatever is to be repeated inside the body.

Now, if you look at pictures, they also contain patterns okay and to draw interesting pictures you need to use repeat statements competently. So, by drawing interesting pictures you are really

getting practice at representing patterns in your programs. And believe me, this is a really important activity, a really important skill while writing programs.

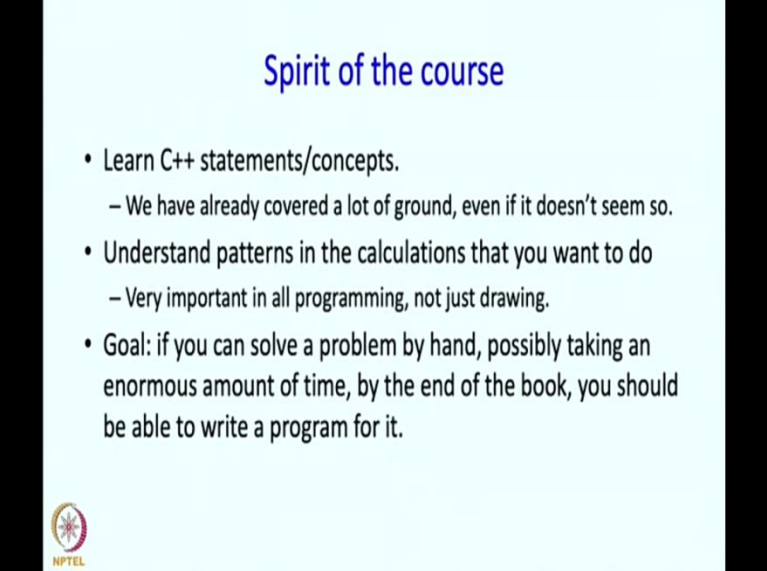
(Refer Slide Time: 2:29)



There are some other reasons as well, so graphical input and output is actually very convenient and useful. This is because often, our programs are about pictorial objects, and graphical output is very important because of the adage that a picture is worth a 1000 words. Sometimes, what you want to tell is much better expressed by a picture and by a picture I do not mean a drawing of a square or a rectangle or a line drawing of some figure or some of a tree or a human face or anything like that, I might mean drawing of a graph, so graphs are very common in science and technology. So what we are learning as we learn picture drawing, is also how we are going to draw graphs which are common in science and technology.


In fact, data visualization is an upcoming area of computer science precisely for this reason that it is very useful in science and engineering. So really you can think of this whole exercise as a very solid step towards understanding this whole issue of data visualization. And of course drawing is a lot of fun. I hope you will agree with me and I certainly think you will agree with me when I show some additional examples okay.

(Refer Slide Time: 3:53)



Spirit of the course

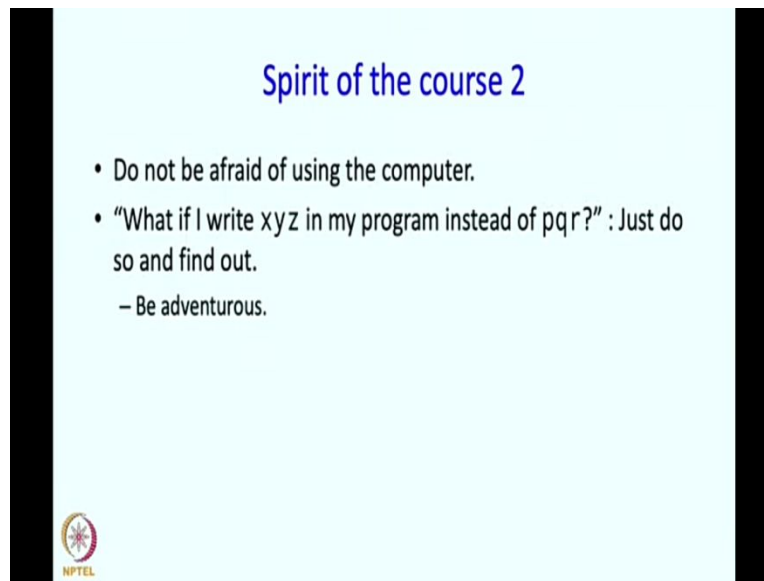
- Learn C++ statements/concepts.
 - We have already covered a lot of ground, even if it doesn't seem so.
- Understand patterns in the calculations that you want to do
 - Very important in all programming, not just drawing.
- Goal: if you can solve a problem by hand, possibly taking an enormous amount of time, by the end of the book, you should be able to write a program for it.



So, what is the spirit of the course? So first of all you have to learn C++ statements and concepts. And I would like to tell you that today, already we have covered a lot of ground, even though it might have hopefully seem like just some fun, but believe me we have done a lot. We have understood patterns, and we have understood that when you program you need to understand patterns. So this is one important point of the entire course - understanding patterns. We have taken a step towards it, but this is a really important part. There can be other types of patterns which we will see a little bit later. And as I pointed out, this is really important in all of programming.

So, what is the goal of the course? Well if you can solve a problem by hand, possibly by taking an enormous amount of time and doing a lot of calculations patiently, by the end of the course or end of reading the book, you should be able to write a program for doing that. And, typically you will see that what you take, what you need hours to calculate, the computer will do in a few seconds. So that is really the goal of the course. Whatever you can do by hand by laborious calculation, you should be able to do using the program.

(Refer Slide Time: 5:47)



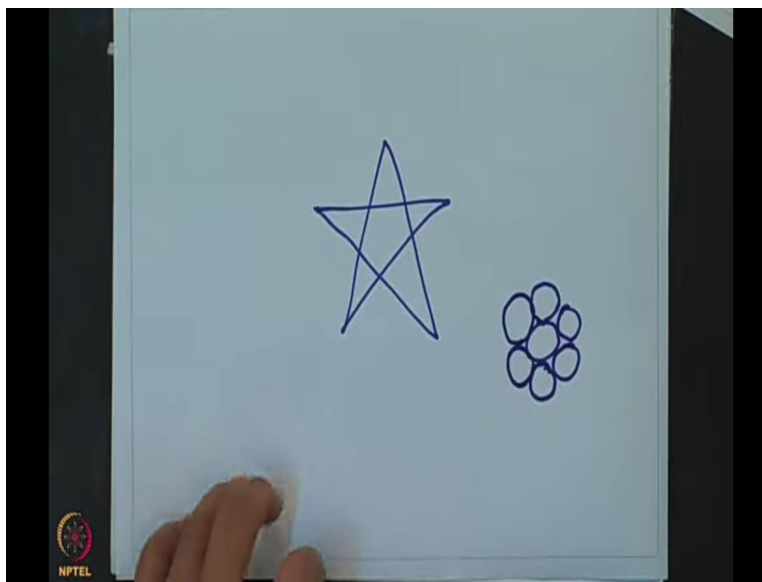

And of course, there may be some things right now, some problems that you do not know how to solve. So for some of those problems, we will also learn some problem solving techniques. One more point about the course is that you have to understand that the computer is your friend. Do not be afraid of using the computer. If you are thinking to yourself “what if I write x y z in my program instead of p q r?”, do not worry about it, do not fret over it. Just do so, just write a program just change that p q r to x y z, compile it and run, and see what happens. The computer is not going to explode, okay?

So there is no cost to being adventurous, there is only gain, you will learn. And a very important thing you have to realize right now is your knowledge is only good if you exercise it and how do you exercise your knowledge? How do you say that I know C++? or how do you say that I know programming? Well you have to write programs, that is the real test. So that is definitely the spirit of the course, programming, practical not theory, well theory and practice okay?. So that really is the spirit of the course. So based on what you have learned so far you can do these exercises.

(Refer Slide Time: 6:51)

Exercises

- Draw a 5 pointed star.
 - Hint: If the turtle starts at a certain point with a certain orientation and returns to the same point with the same orientation, it must have turned totally through a multiple of 360 degrees
- Draw a 7 pointed star. How many different 7 pointed stars can you have?
- Draw 7 identical circles, with 6 touching the central circle. Circle = polygon of large no of sides, say 360.
- Draw 4x4 array of tiles slightly separated from each other.

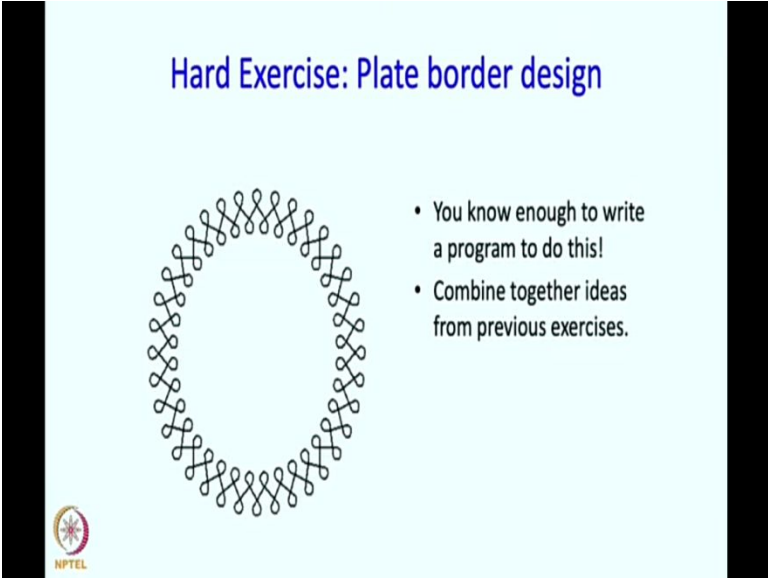


So for example, the first exercise asks you to draw a 5 pointed star. So let me tell you what I mean there. So this is the star that I am talking about. So it will be drawn using 5 lines and some turning. The question is how much turning we have to do and the hint tells you how to do that. Now, why stop at stars with 5 corners? Why not draw 7 pointed stars? And you could also ask is there only one kind of 7 pointed star? or are there several kind of 7 pointed stars? You should be able to figure this out and you should be also able to draw them.

Now, you can also draw circles, because what is the circle after all, a circle is an n-sided polygon in which n goes to infinity and the side length becomes really small so that is how a circle is drawn in a computer. So go ahead and draw an n-sided polygon where n is large and you will see that it looks like a circle.

Then this problem asks you to draw 7 identical circles such that they touch each other. So the expected figure is something like this. My drawing is not great, but I intended these circles to touch each other. So you can do this. And then the last one is about drawing array of tiles which are slightly separated from each other.

(Refer Slide Time: 8:27)



The slide features a light blue background with a black border. At the top center, the title "Hard Exercise: Plate border design" is written in blue. Below the title is a circular border composed of interlocking, teardrop-shaped elements. To the right of the border, there is a bulleted list of two hints. In the bottom left corner, there is a small NPTEL logo.

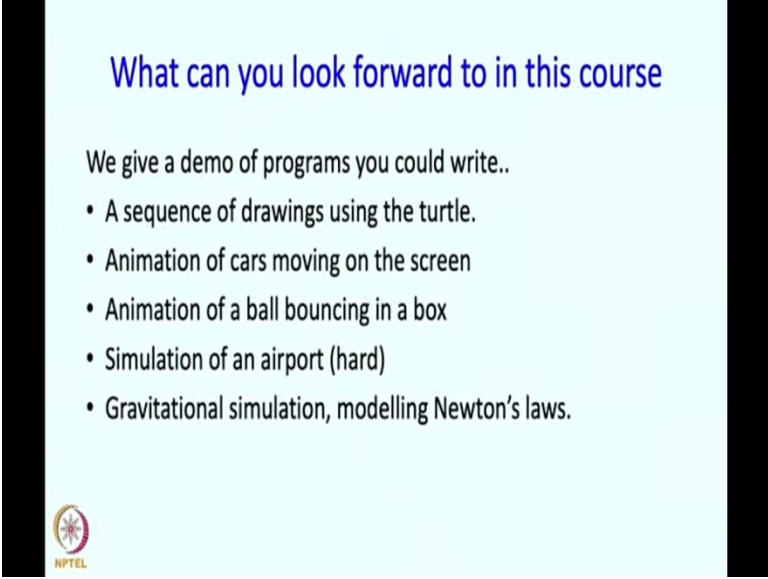
Hard Exercise: Plate border design

- You know enough to write a program to do this!
- Combine together ideas from previous exercises.

NPTEL

This is a slightly harder exercise, you can think of this design as the design that might appear on the border of a plate. So use your turtle to draw this. Actually you know enough program to do this okay. So basically, you are going to have to combine the ideas that you will have learnt in the previous exercises. So one important hint: do not give up and one more important hint: try out whatever ideas come to your mind try out. You will see that some of them work. In fact you will see even if the ideas do not work they may produce an interesting drawing. They may produce a drawing which is even nicer than this, but anyway persist and get to this drawing.

(Refer Slide Time: 9:40)



The slide has a light blue background with a black border. The title is in blue text. Below the title is a list of five items, each preceded by a bullet point. In the bottom left corner, there is a small circular logo with a star-like pattern and the text 'NPTEL' below it.

What can you look forward to in this course

We give a demo of programs you could write..

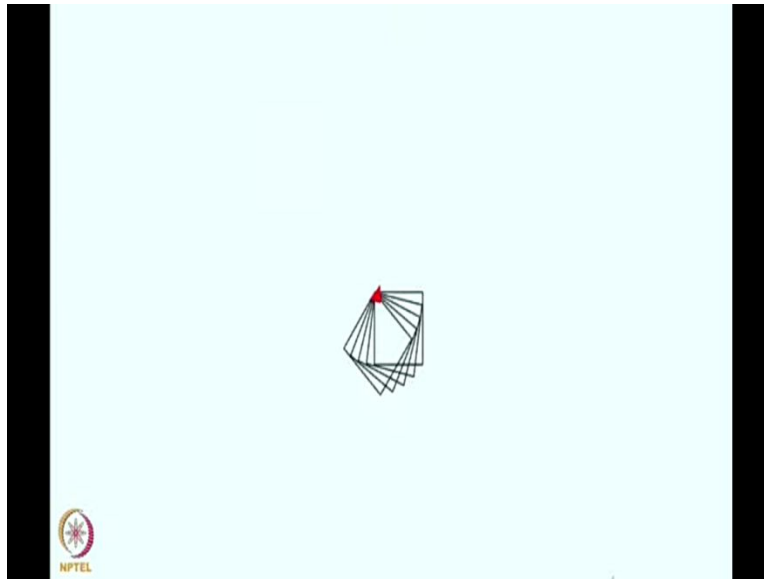
- A sequence of drawings using the turtle.
- Animation of cars moving on the screen
- Animation of a ball bouncing in a box
- Simulation of an airport (hard)
- Gravitational simulation, modelling Newton's laws.

NPTEL

So I am going to conclude this lecture at this time, but before that I am going to give you a demo which shows you what kind of programs you can write at the end of the course. Some of these programs are slightly more advanced than that, but only slightly more advanced. In any case they are discussed in the book. So here are some of the things, so we will show you a sequence of drawings, interesting drawings that the turtle can draw and you will learn these in the very first few lectures. You will see an animation of cars moving on the screen, you will see animation of a bouncing ball, then you will see a simulation of an airport, you will see some drawings of trees, so trees and some abstract diagrams, and you will see simulation of planetary movements, planets moving around the stars.

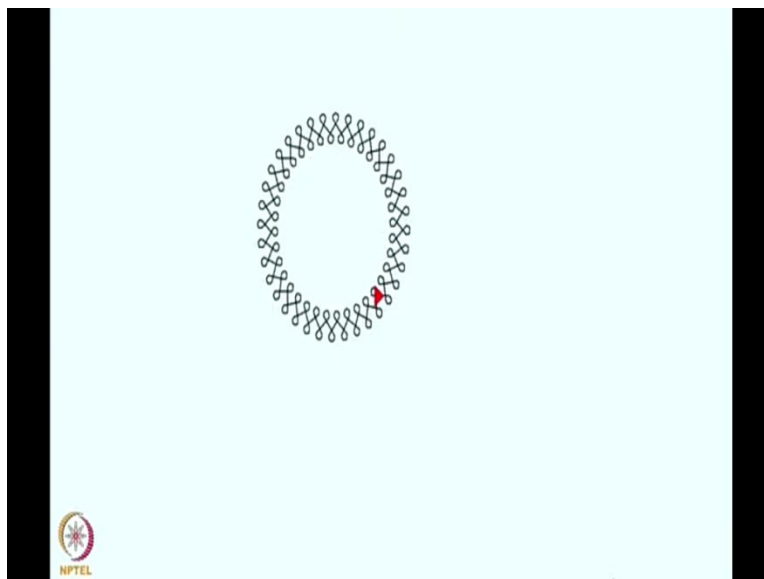
So this has been done by actually modeling Newton's laws of motion, and Newton's law of gravitation, this is kind of the code that an astronomer would write. Well the astronomer would write slightly more sophisticated code, but this is actually somewhat sophisticated. But you will see that it is actually not that complicated. You know enough science and math to be able to understand these codes, and these codes most of these codes are really very small, 20, 30 lines, only a couple of them are slightly bigger than 100 lines. So we will stop over here, but I will show you a demo and then the lecture will end, so hold on okay, so let us do this.

(Refer Slide Time: 11:02)



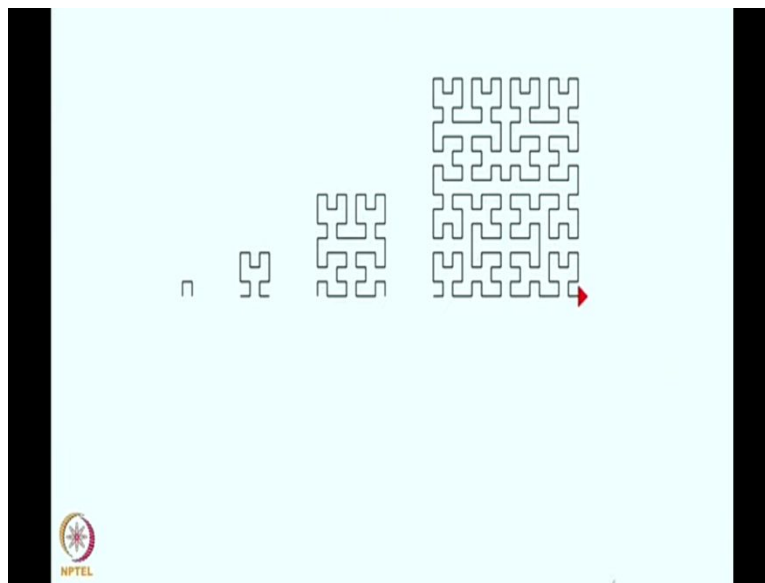
So, I am going to click this video, so the turtle has appeared and it is going to wait a little and then it is going to start and it will draw the square first, so you can treat this as an exercise. How will you draw such a pattern, okay? These expanding circles by the way, have also been drawn using simple cpp, so again that would be a good challenge for you to draw after a little bit of this course is over.

(Refer Slide Time: 11:28)



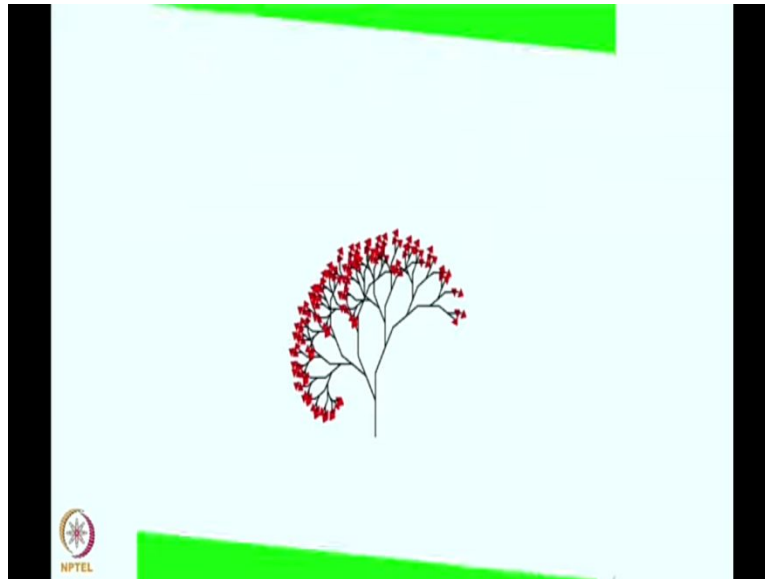
So here is the turtle executing the program which will draw the plate design okay. Now, I should tell you that in this plate design that the pattern repeats 36 times, but when you write the program, you should really write the program so that if I want the pattern to repeat 100 times, you should be able to do that okay, so here is one use of computers - so a computer can generalize. So if you, so for a human being, you may sort of take a lot of trouble and you may draw the single pattern, but once you write a program, you can generalize it and you can write the program so that it draws pattern with 100 such repeated figures.

(Refer Slide Time: 12:26)



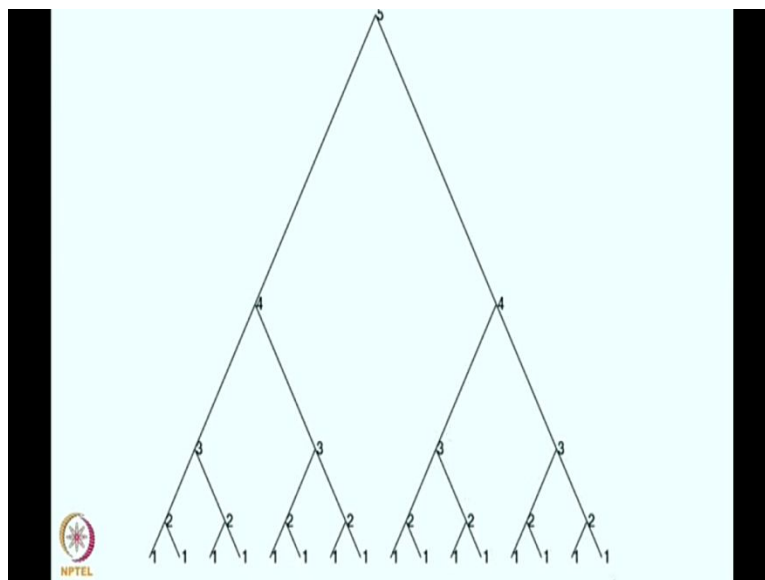
So this is a curve, which is called a 'Hilbert space filling curve'. You will see that the first pattern appears in the second, the second appears in the third, the third appears in the fourth, and there is not going to be any fifth pattern because the fifth pattern will become really big, but this is an interesting figure named after the German mathematician Hilbert.

(Refer Slide Time: 12:56)



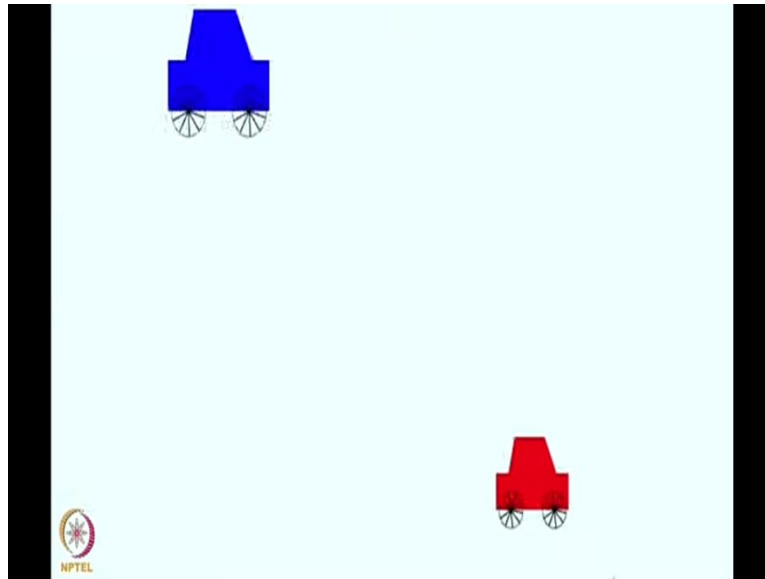
So here, the turtle is actually making copies of itself and then each of those copies are doing the part of the drawing, so I like to call this a gulmohar tree, because the color is really like the gulmohar flowers which appear in summer in India, and I think this is a really pretty tree.

(Refer Slide Time: 13:27)



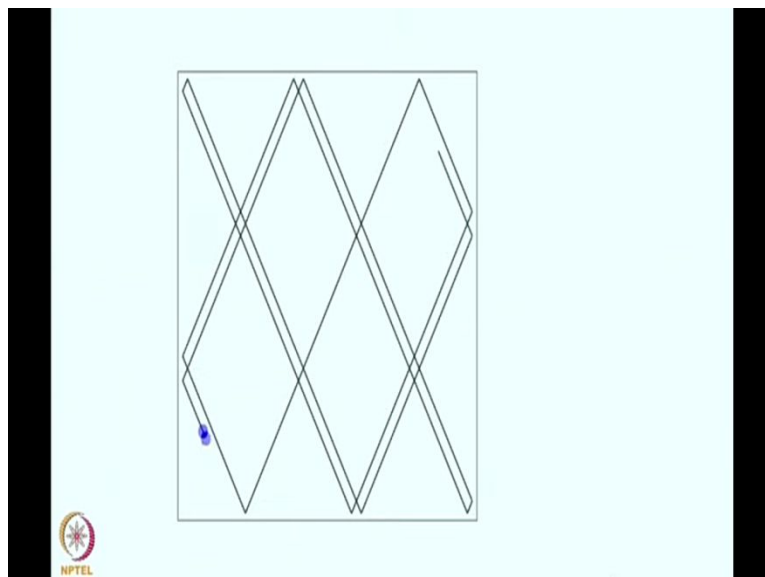
So this is a stylized diagram also called a tree in computer science, so you can not only draw diagrams, but you can also put numbers and that is what that example shows.

(Refer Slide Time: 13:40)



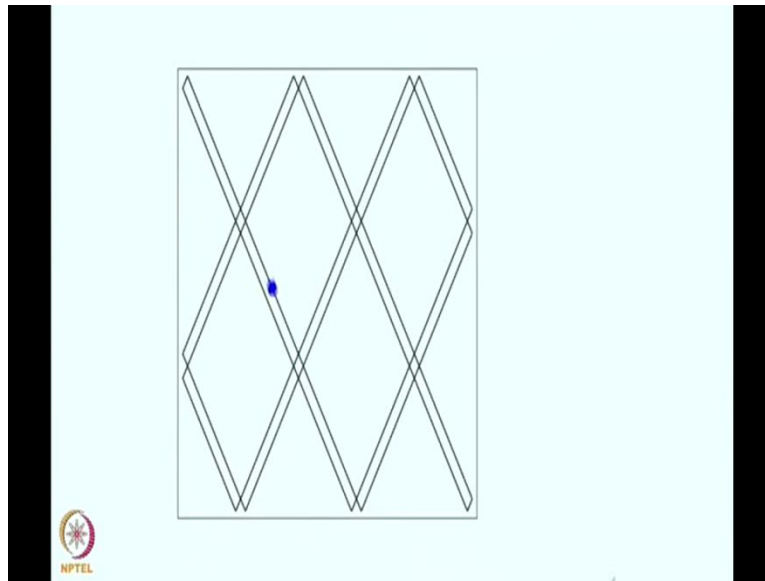
Here is a little bit more complex drawing and in this we have these two cars. So they really are made up of several small elements, so they are made up of circles and they are made up of lines and they are made up of polygons. And it takes a little bit of coordination to make sure that all these elements are moving the way you want them to, but that is what programs are for. You can actually write program which do these things.

(Refer Slide Time: 14:12)



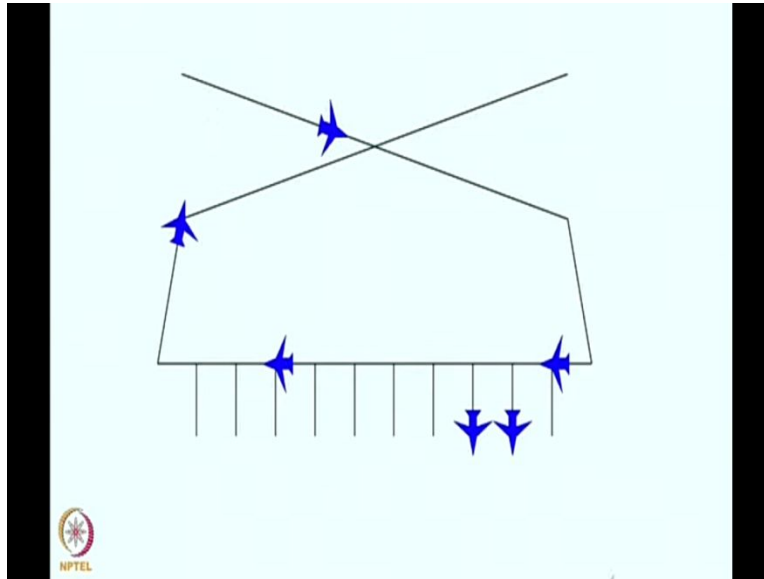
Here is a kind of Physics experiment if you will, you have a ball which is bouncing inside a rectangular region and it is obeying the rule that the angle of bounce back is equal to the angle at which it approaches the ball and of course it has a pen, so it is tracing the path, it is showing the path that it traces. The pen is at the centre so the line is not going all the way till the end of the ball okay, all the way till the ball.

(Refer Slide Time: 14:58)



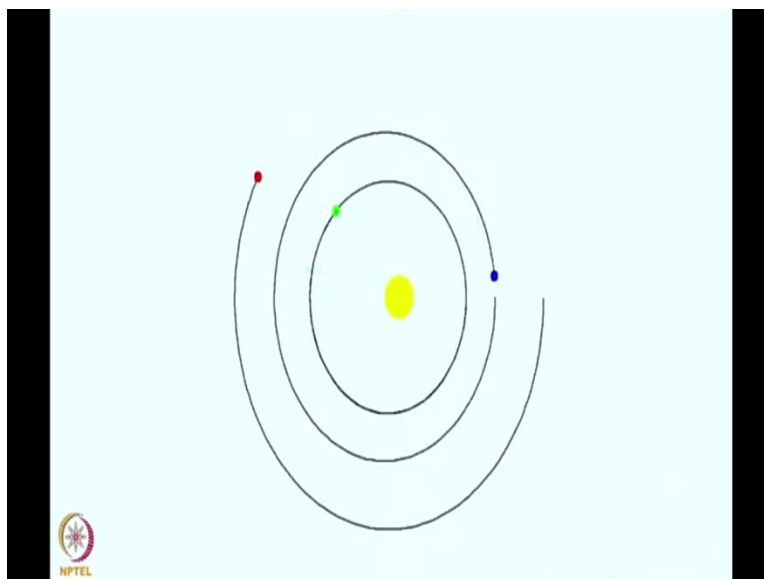
Now, you will see something interesting happen as it makes this drawing and that is because the velocities the x and y coordinates of the velocities have small multiples with respect to the sides of the square. Anyways now what is going to happen is it will retrace that same pattern.

(Refer Slide Time: 15:08)



Here is probably the most complicated example in this, and that is the simulation of an airport. So you will see that the aircraft actually keep some distance from them. At one point, they seemed as if they were too close that is because I have made the picture of the aircraft larger than what it actually is. So you can see that on any line segment there is only one aircraft at a time.

(Refer Slide Time: 15:46)



So this is the planetary simulation and this is what will need some sophisticated physics, actually not that sophisticated, nothing that you did not learn in standards 11th and 12th. With that we

conclude this first lecture of the first week of An Introduction to Programming through C++.

Thank you.