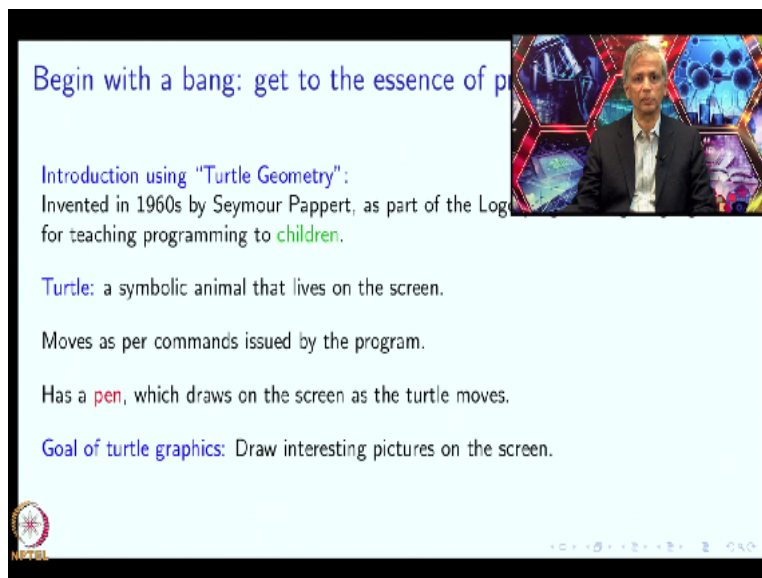


Design and Pedagogy of The Introductory Programming Course
Prof. Abhiram G. Ranade
Department of Compute Science and Engineering
Indian Institute of Technology - Bombay

Lecture – 14
Pedagogy.2: A Quick Tour of the Course - 1

Hello and welcome back. We have been discussing the pedagogy of the course and we have just discussed how many lectures we are going to spend on each topic and now I want to take a quick tour through the material and make comments about how to teach different topics.

(Refer Slide Time: 00:44)



Begin with a bang: get to the essence of p

Introduction using "Turtle Geometry":
Invented in 1960s by Seymour Pappert, as part of the Logo
for teaching programming to children.

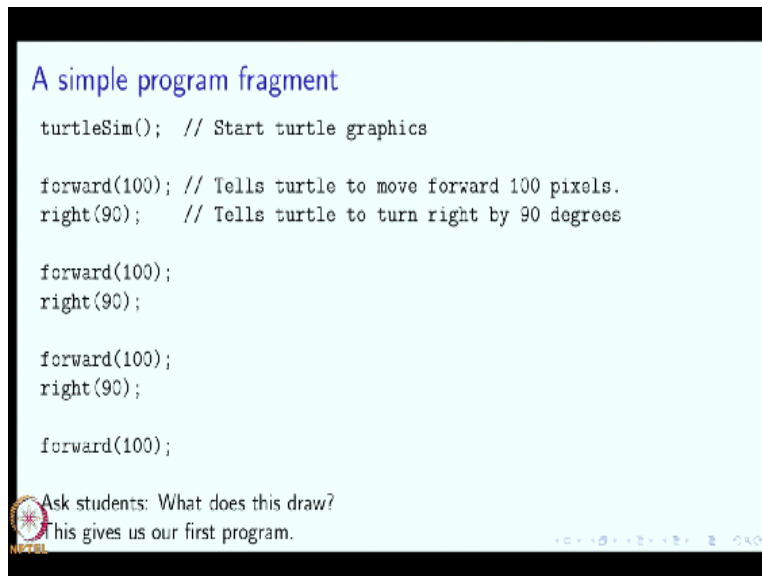
Turtle: a symbolic animal that lives on the screen.
Moves as per commands issued by the program.
Has a pen, which draws on the screen as the turtle moves.
Goal of turtle graphics: Draw interesting pictures on the screen.

The first topic is how to begin. So I think it is really important, if possible, to begin with the bang and somehow get to the essence of programming on the first day because as we argued earlier if you do that, you will get your students excited, you will get your students motivated and that motivation and that excitement will stay through the course. So I recommend an introduction using Turtle Geometry.

So this Turtle Geometry was invented in the 1960s by Seymour Pappert as a part of the Logo programming language. I should say Seymour Pappert and his collaborators and this was for the teaching programming to children. So a turtle in Turtle Geometry is a symbolic animal that lives on the screen, okay and it moves as per the commands given by the program.

It has a pen which draws on the screen as the turtle moves. And the goal of turtle graphics is to draw interesting pictures on the screen, okay. Not compute but draw interesting pictures. You will see why that is not an easy undertaking but yet we can already guess that it should excite the student, excite young children.

(Refer Slide Time: 02:10)



```
A simple program fragment

turtleSim(); // Start turtle graphics

forward(100); // Tells turtle to move forward 100 pixels.
right(90);    // Tells turtle to turn right by 90 degrees

forward(100);
right(90);

forward(100);
right(90);

forward(100);

Ask students: What does this draw?
This gives us our first program.
```

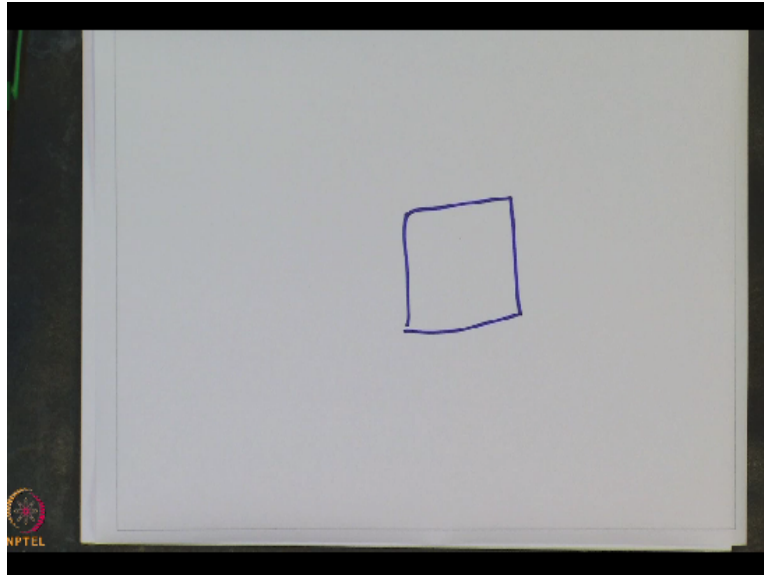
So let me begin by showing you a simple program fragment, the hello word program if you will of turtle graphics. So the first statement in this program turtleSim is a statement which causes the computer to star up turtle graphics. So this will create a window on the screen, okay, the window in which the graphics will happen. Once this happens, you will see a small red triangle come up and that triangle will symbolically represent a turtle.

Forward100 is the next statement and this statement tells the turtle to move forward 100 pixels. Right90 tells the turtle to turn right by 90 degrees. While explaining this you could say that instead of write 90, if I write, write 45, the turtle will turn by 45 degrees. So you get to tell the turtle how much to turn. So the next is forward100 again. After that, it is right90 again, forward100, right90, forward100.

So let us pause for a second. You can tell your students and try to think of what picture the turtle has drawn. Well, let us try that. So suppose this is our turtle and it is pointing in this direction and the first command it sees is forward100. So it moves forward 100 pixels. After that the next

command it gets is right90. So it turns right 90 degrees. Then it moves forward 100 pixels again, again turns right 90 degrees, again moves 100 pixels, again turns right 90 degrees and again moves forward 100 pixels. What has, has it done? Clearly it has drawn the square, okay.

(Refer Slide Time: 03:59)



So this is how you draw a square and you can expect students to understand this very fast and if you show them the program and execute the program for them as we will in a minute, you will see that the students learn write away, okay. So you ask the students what does this draws? So this gives our first program.

(Refer Slide Time: 04:44)

```
The full day 1 program

#include <simplecpp>
main_program{
    turtleSim();
    forward(100); wait(0.5); right(90); wait(0.5);
    forward(100); wait(0.5); right(90); wait(0.5);
    forward(100); wait(0.5); right(90); wait(0.5);
    forward(100);
    wait(5);
}
```

- ▶ <simplecpp>: For graphics functionality. Also includes "using namespace std;" and <iostream>.
- ▶ main_program: C++ preprocessor macro. Gets translated to int main()
- ▶ Students must understand namespaces, iostreams, int main, but not on day 1.
- ▶ wait : So that we have time to see..

Well, the first program actually is a little bit more complicated because there are additional

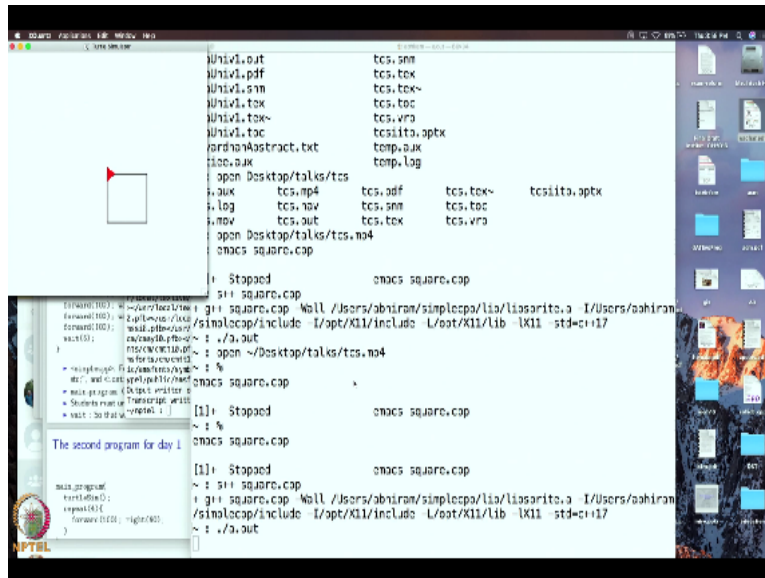
things in it. So there are these waiting statements, okay. So you have to make the turtle wait because otherwise, the whole thing will happen so fast that you will not get a chance to see it and then you need that line `simplecpp` because you want to tell the computer that look I want to use the graphics functionality.

Now all of us teachers know that when we write a C++ or C program, we need some preamble and for example in C++, it is customary to include the header file `iostream` and also say may be that we are using the `std` namespace. Now this is very difficult to explain to the students on the first day. So we are going to do this inclusion as a part of that includes `simplecpp` itself. So we do not, we can spare these gory details to the student on the very first day.

Main program is also a preprocessor macro just as we talked about the preprocessor macro `repeat`. Now this gets translated into `int main` and again, the motivation for this is that `int main ()` is really complicated to explain on the first day. So instead of that, let us hide it and we will explain about `int main` when we talk about functions. So do not, do not, I mean, this is, this is a detail, this is an uninteresting detail right now.

So let us not clutter up our teaching with these uninteresting details, okay. So yes, students first understand namespaces, `iostreams`, `int main`, but they do not have to understand it on the first day, okay. So let us stop over here. Oh, yes, by the way, the `wait` is going to be put there so that students have time to see what is, what the turtle is doing. Okay, so now I am going to use my computer and I am going to show you how to compile and execute that program.

(Refer Slide Time: 06:54)

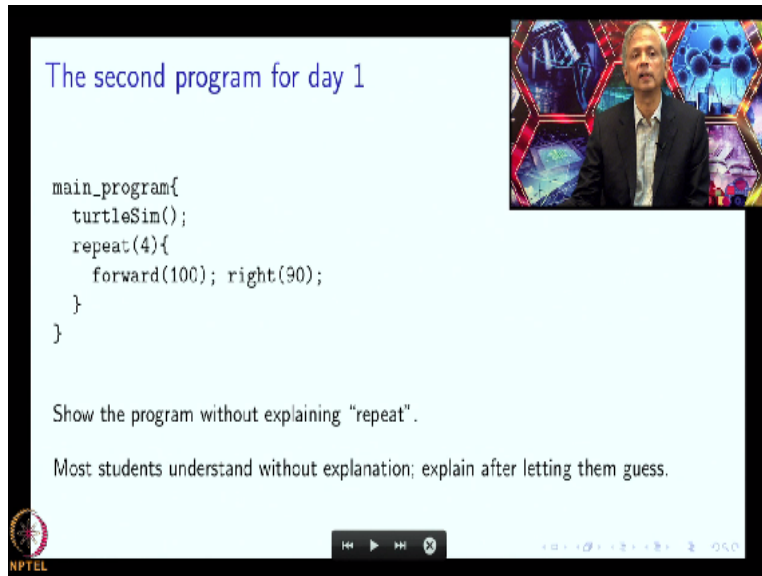


So I have an editor called EMACS which I am going to use but you could have used any other editor as well. So in my EMACS editor, I have already typed in this program. So as you saw, I had written include simplecpp, turtleSim and all of that. Let me increase the font a little bit, okay, yes. So I have this program. You can see, I have put in wait0.5 after forward and right so that the turtle waits and at the end, I have put in a wait of 5 seconds.

So this program we typed, we saved it and now we are going to compile it. So the compiler script is S++ instead of G++ and now I am going to say S++ find name, so square.cpp, so it is, it has now compiled. As you can see when I use S++, it actually causes the G++ compiler with various options. So the student does not need to know the options but they are being shown here just in case somebody is curious.

Now I have to execute this. So I can type ./a.out. So that creates this turtle. It executes that square drawing program and then it waits for 5 seconds and disappears, okay. Alright. So let us go back to the lecture. Okay. So this is the full day 1 program that we talked about and we just executed a minute ago.

(Refer Slide Time: 08:32)



The second program for day 1

```
main_program{
  turtleSim();
  repeat(4){
    forward(100); right(90);
  }
}
```

Show the program without explaining "repeat".

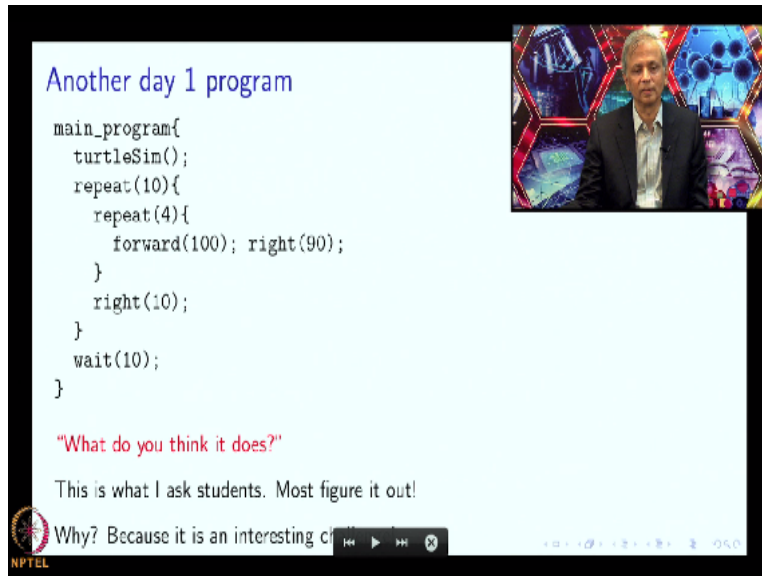
Most students understand without explanation; explain after letting them guess.

NPTEL

So here is the, the second program for the first day. So here I have everything except that instead of typing in forward100, right90 4 times, I have put them inside a repeat. So what I do when I teach this course is I show students this program, okay. Without explaining what repeat means and I ask them to guess. What do you think this could mean? Most students understand without explanation.

After all what else could it mean? It says repeat 4. It cannot mean repeat 15 times or repeat has to have some meaning. So it had better mean repeat whatever follows 4 times. So students will guess that it means repeat a sequence of command forward100, right90 4 times. So this program will do pretty much the same thing as the last program except that it is much more compact.

(Refer Slide Time: 09:37)



Another day 1 program

```
main_program{
  turtleSim();
  repeat(10){
    repeat(4){
      forward(100); right(90);
    }
    right(10);
  }
  wait(10);
}
```

"What do you think it does?"

This is what I ask students. Most figure it out!

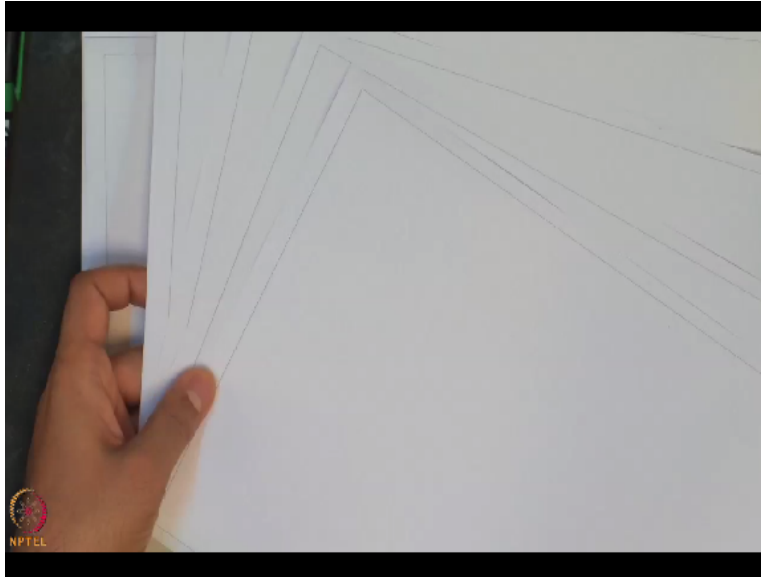
Why? Because it is an interesting c

NPTEL

Here is another day 1 program. So this time, we have repeat 4 which contains forward100 and right90, that is inside another repeat 10. So we are going to draw a square, the repeat 4 100, forward100, right90 is going to draw a square and we are going to repeat it 10 times. So are we going to draw the same square again and again? Well, no. We are going to turn right by 10 degrees.

So what is going to happen is that if we are drawing squares, we first draw one square, then we turn and then we draw a square, sorry, we first draw one square, then we turn, draw a square at a certain angle, then we draw another square, then we draw another square and we do this at different angles.

(Refer Slide Time: 10:34)



So that is how this program is going to work, okay. Now I, I show this program on my slides or on my board and I ask the students, what do you think it does? I do not explain to them right away. Now believe me students think about it and students explain it, quite correctly. Many students do, not everyone probably. But many students know exactly what is going on. Why is that?

I think that is because repeat is so intuitive, if I put a 1 repeat inside another repeat, then students have a clue that what is inside has to be repeat as many times, repeated as many times as the outer repeat, okay. I explain this of course, but only after they have a chance to figure it out and they do figure it out, okay. And they do work hard because they are interested. Look what could the turtle be doing here, that is what they think, okay.

(Refer Slide Time: 11:43)

What have students learnt on day 1?

1. Control flow
2. Elementary iteration, including nested iteration
3. Basic ideas of syntax, spaces, indentation
4. Importance of observing patterns in what is to be accomplished, and expressing them using repeat
Do not write 100 statements to draw a 100 sided polygon!
Very important activity while designing programs!

Essence of programming?

Last activity for day 1: Show movie of what is possible using graphics.

NPTEL

Alright, so this is more or less what I talk about in the first day. Because on the first day, there are also maybe other things to talk about like what is the textbook, what are the hours, where is the lab and all of that? So there is not that much time but there is enough time to say all these things. So what have the students learnt? Well, let us next talk. First, they have learnt control flow.

Well, they will not learn it on their own, you have to say that just as read from left to right, top to bottom, a computer also executes statements top to bottom, left to right, okay. And then they have learned that well it is not always top to bottom but there can be a repeat in which case some statements might get executed several times and in fact, there can be one repeat inside another repeat.

So see what they are learning on the first day. They are learning iteration and nested iteration. So that is pretty good progress. They have, they will learn the basic idea of syntax, spaces, indentation and you should tell them also but notice that the way we have structured the lesson, the focus will not be on syntax. It will not be on that you should have a space in-between one word and another, okay.

But it will be like I want to do this and I want to write the program and my program should look good and what is the natural way of writing that program. So a natural way of writing a program

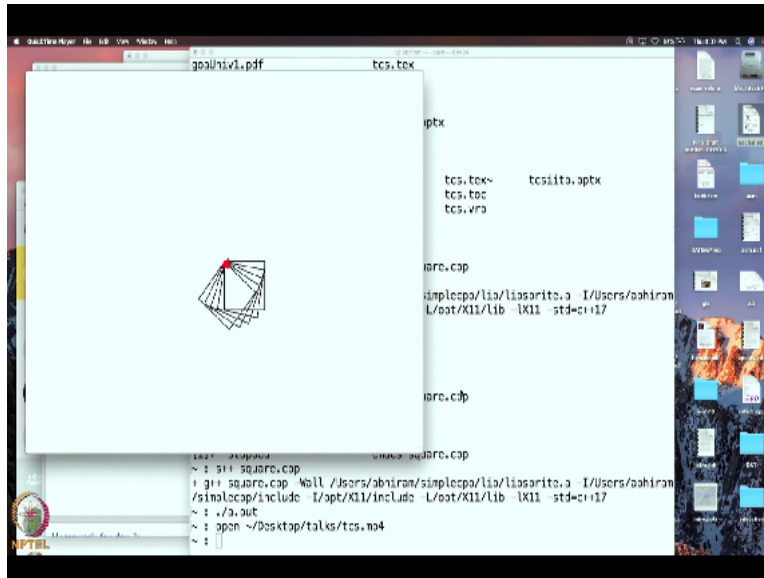
will be to put indentations. So you do explain the indentation but the indentation is sort of natural because they are following along with the logic. Then there is something else that they also understand, they should understand or at least they have an, an opportunity to understand which is of importance of observing patterns in what is to be accomplished and expressing them using the repeat, okay.

In other words, if I want to draw a 100 sided polygon, I should not be writing 100 statements or rather I should not be writing 100 forward and 100 right statements. That is just too painful. What should I do instead? I should realize that there is some repetition, there is a pattern and I should express that pattern in my program. That is a really important idea. It is a very important activity, recognizing patterns and expressing them in your programs is a really important activity while designing programs.

I will go as far as saying that this is an essence of programming. This is an extremely important idea in writing a program. Not that you should just know what a repeat statement is but you should know this idea that the purpose of a looping statement is to express a pattern and not have to write so many statements.

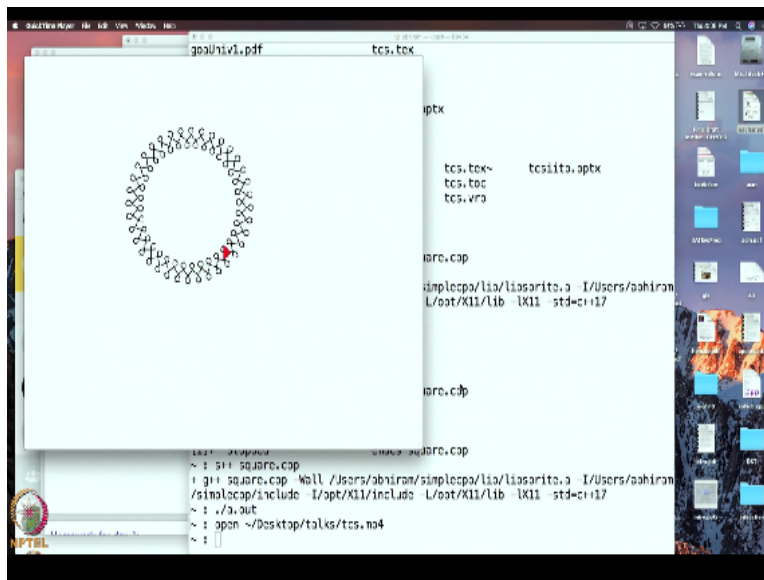
So whatever patterns come up in your computation, should get expressed in your program. So this is the starting point of writing programs or rather I should say writing good programs. The last activity for day 1 is to show students a movie of what is possible using graphics, okay. So let me do that now.

(Refer Slide Time: 15:25)



Okay, so we will talk about homeworks but let me first talk about, show you the movie. Okay, so I am back to my computer and I am now going to open the movie. So let me get it running. So there is that turtle and it will start moving. In fact, it will draw that nested square pattern, okay. So as you saw it is turning and then drawing another square, okay. So that is the first, the first and a program that I want to show you.

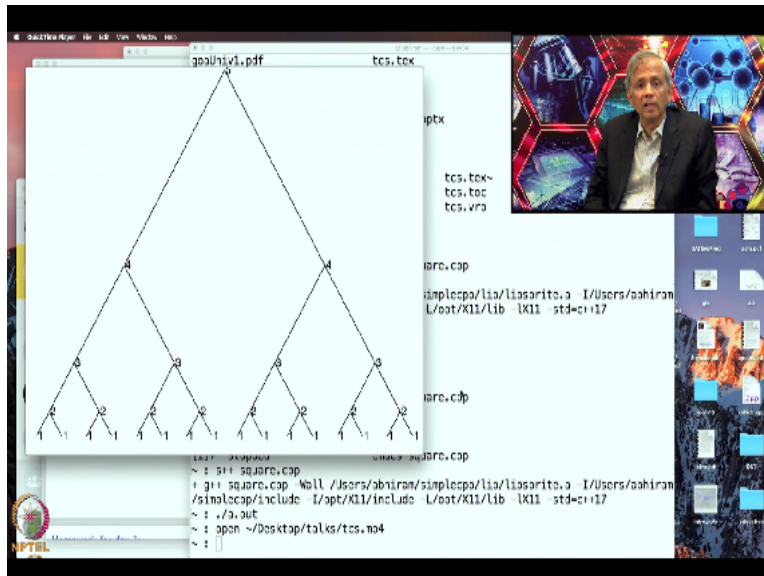
(Refer Slide Time: 16:18)



After that the turtle is going to draw a pattern which you might think of as being used for decorating plates. So this is the kind of a pattern which might appear on the border of a plate, okay. Now it turns out that this pattern can be drawn using some 10-15 lines. It just consists of 3 nested repeat statements and of course, forward and right statements. So the picture of this

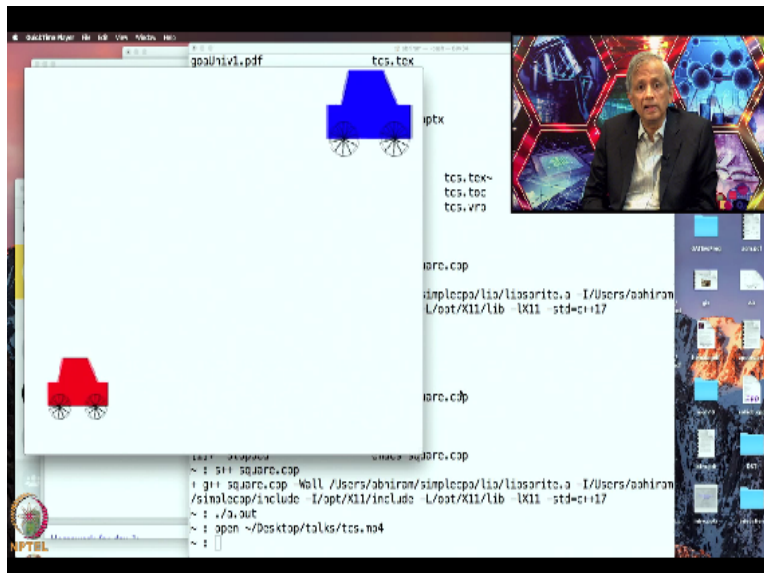
recursive picture that is being drawn. This recursive picture is drawing out a tree. So the code for this is also maybe 10-15 lines. Of course, there is a function, a recursive function which is being called and you can see the turtle is drawing the code but as it draws the code, it leaves behind copies of it. The copies are just there so that they appear like flowers or leaves, okay. But it looks like a pretty realistic tree.

(Refer Slide Time: 18:12)



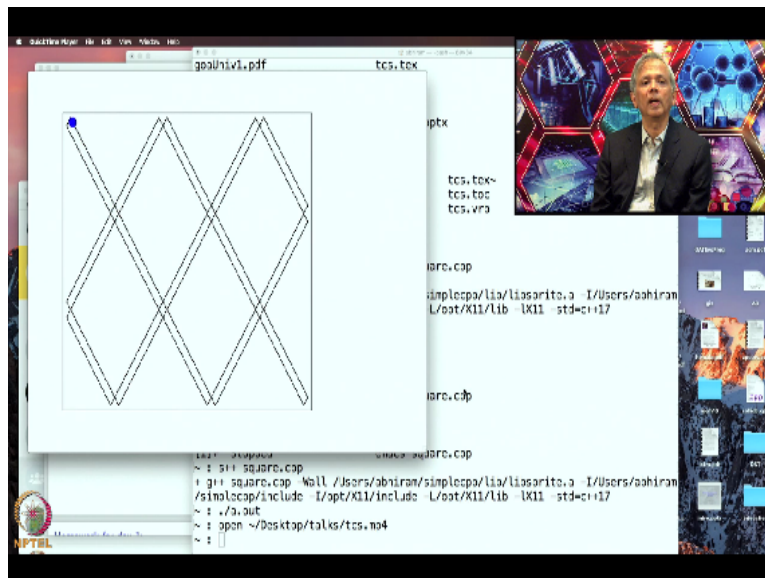
Here is a tree which is again drawn using recursion, recursion turns out to be an important idea and it can be expressed nicely. So many of these pictures can be drawn with very small amount of code.

(Refer Slide Time: 18:24)



This is, this is used, this is done using some code which comes up towards the end of course. This is something that uses inheritance. It is a little bit complicated but students will be able to do it if they even read the, the chapters on inheritance by themselves, okay. So you may notice that the wheels are turning just the right amount.

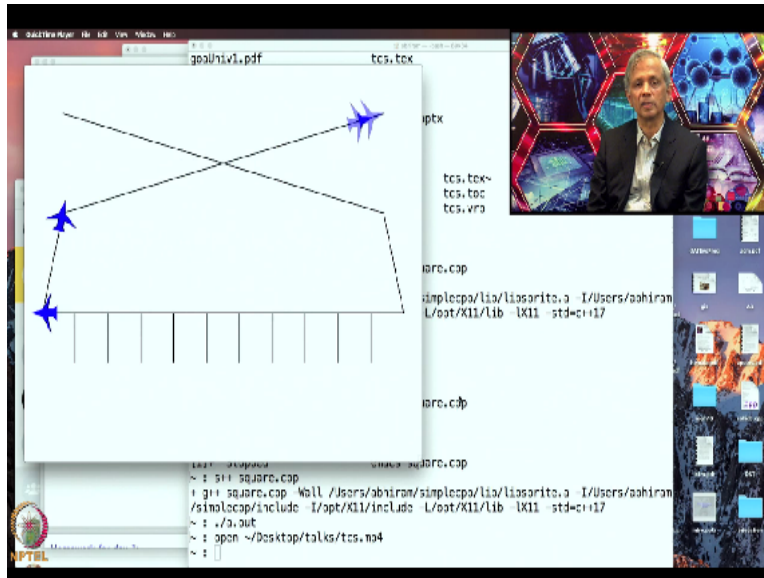
(Refer Slide Time: 18:58)



This is something that comes up early actually. So this just requires a conditional statement and a repeat is enough and this does not use turtle graphics but uses a different kind of graphics and what it is doing as you can see is it is simulating a ball bouncing. So a bouncing ball is kind of a very simple representation of a mechanical system and the point of this is to draw a nice animation but also understand that many mechanical systems can be pictorially simulated.

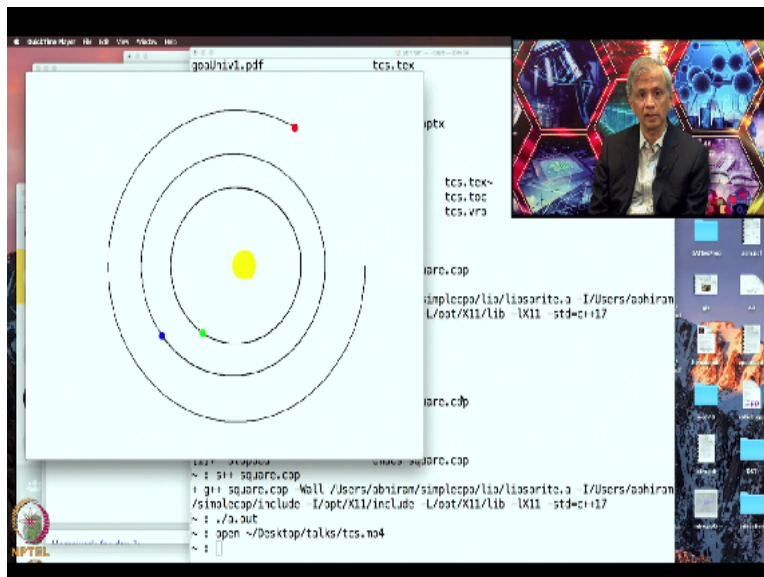
And that simulation is in principle can be useful and there is something interesting going on over here. Somehow or the other, the ball is going to now retrace its path.

(Refer Slide Time: 19:54)



This is the most complicated program in this demo. It consists of this simulation of aircraft. The aircraft are shown big so that you can see them but actually the way the program is written; they do not actually intersect. In fact, if you look at it carefully, the aircraft are kept far apart from each other, especially when they are going fast. So this is meant to be a real simulation of a two-runway airport, well not a real simulation, sort of a quasi-real simulation.

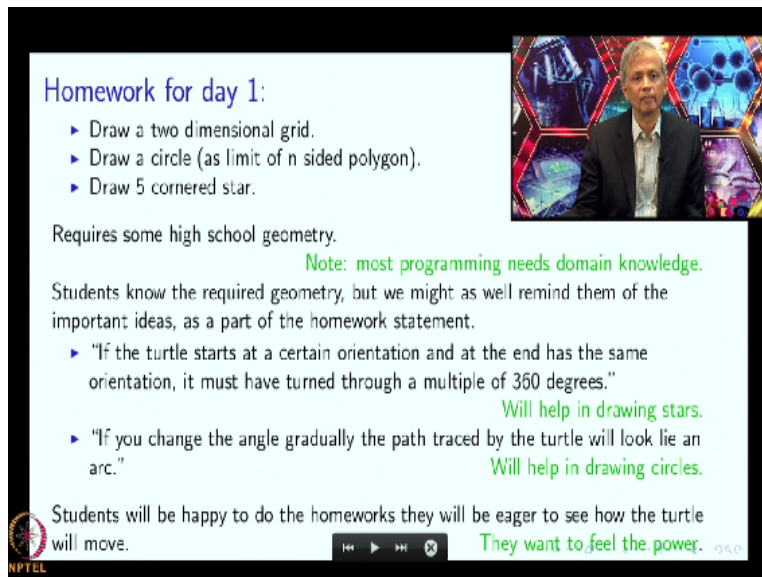
(Refer Slide Time: 20:32)



This is going to simulate our solar system and it actually uses Newton's gravitational law and calculates forces according to that, okay. So that is it. So this, these are some of the things that you should tell the student you can do if you persist with the course and I think that will get the students excited. Okay, so we were talking about what students have learnt on day 1 and then we

saw this demo and so the last activity was the demo for that day.

(Refer Slide Time: 21:25)



Homework for day 1:

- ▶ Draw a two dimensional grid.
- ▶ Draw a circle (as limit of n sided polygon).
- ▶ Draw 5 cornered star.

Requires some high school geometry.

Note: most programming needs domain knowledge.

Students know the required geometry, but we might as well remind them of the important ideas, as a part of the homework statement.

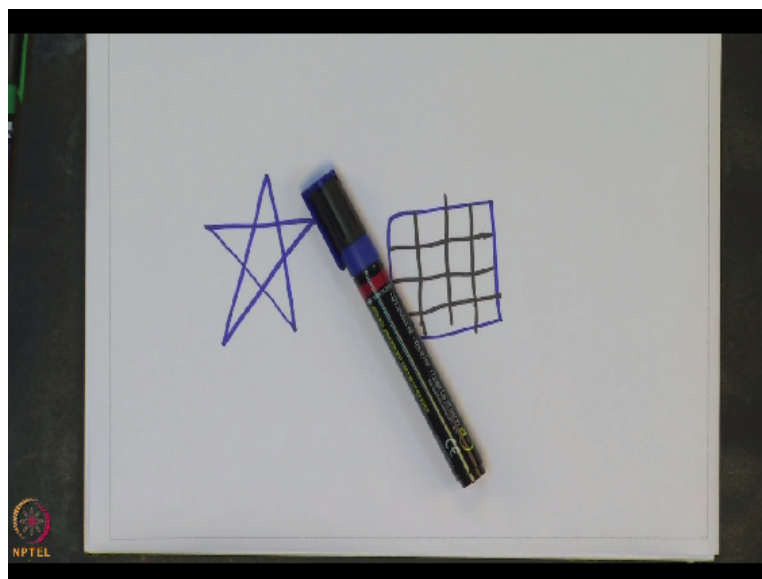
- ▶ "If the turtle starts at a certain orientation and at the end has the same orientation, it must have turned through a multiple of 360 degrees."
Will help in drawing stars.
- ▶ "If you change the angle gradually the path traced by the turtle will look lie an arc."
Will help in drawing circles.

Students will be happy to do the homeworks they will be eager to see how the turtle will move.
They want to feel the power.

NPTEL

But we should also give some homework. So what homeworks can we give? Well, of course, we can say if we can draw a square, can you draw a polygon. But we can do more. Can you draw a 2-dimensional grid, okay? So by that I mean, simply something like this.

(Refer Slide Time: 21:44)



And of course in this, the turtle will have to go over some of the lines it has already drawn, okay. Or maybe, there are commands for raising the pen and lowering the pen. So these commands can be supplied as a part of the statement of this or they could have been discussed in the class. I think if you see the slides for the book, they are discussed in, expected to be discussed in the

class on the first day.

Maybe you can ask or you can show them how to draw a circle because they have learnt, our students have learnt in mathematics that a circle is a limit of an n -sided polygon as n tends to infinity. So how do you draw a circle. Well, you draw an n -sided polygon where n is large as simple as that. So for practical purposes, if you draw 100-sided polygon on a computer, it looks just like a circle.

So students will have lot of fun drawing circles, drawing circles of various, various sizes and actually one of idea is in that plate drawing was part of it were arches of a circle which you got drawn. How do you draw a 5 cornered star? Okay by this, I mean simply something like this. This star that we draw without lifting the pen, okay. Now in this problem, as well as in that plate design problem, there is an interesting geometric principle at play.

And that interesting principle is that if I start with my turtle pointing in this direction, I draw a complicated drawing and eventually I come back and again I am pointing in this direction. So how much total turning have I done? So I have done turning equal to 360 degrees multiplied by some integer. So just alert your students of this fact and then they will be able to reason out everything else.

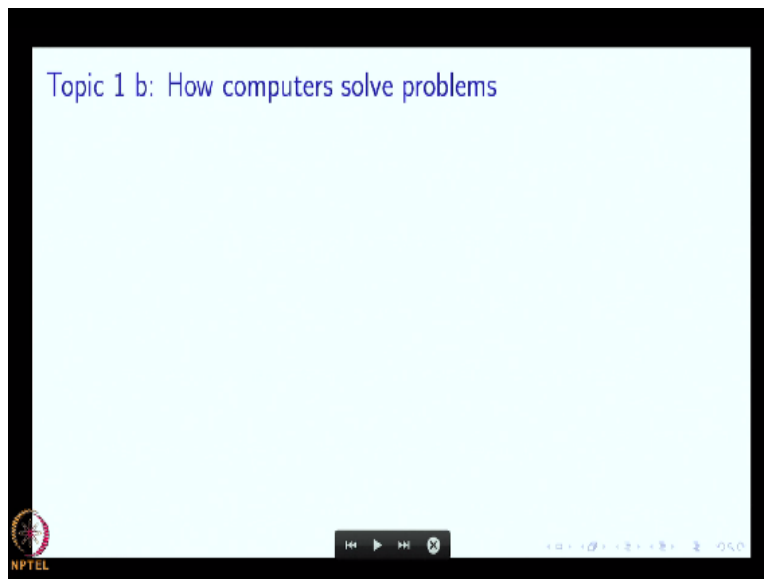
So they will see that look I am in the star, I am, I am going to turn by 2 times 360 and over the entire drawing, I am turning 5 times, so then my turning angle should be what? $720/5$. So this, all these things require some high school geometry, okay. But most programming needs some domain knowledge. Programming is not something to be done by itself. It has to be, we write a program about something, about some subject and that subject has to be understood by us.

Now here geometry is something that students have learnt through high school and also through junior college and therefore, they know that and of course, let us not assume or let us not scold our students if they do not know it, but let us just be gentle and let us remind them a little bit but yes, they already know most of it and so we will have to do only a small amount of reminding. But the point is that this is something that we need to understand.

Whenever we write a program, it is going to be about a domain and they need to have that domain knowledge. So, yes, so we need to remind them of the important ideas and this can be done as a part of the homework statement. So what I said a minute ago, that it has to have turned a multiple of 360 degrees could be given as a part of the homework statement, okay. So it will help in drawing stars, it will help in drawing that plate design that I showed, okay and how do you draw a circle?

Well you change the angle only gradually. Now my experiences that students are very happy to do these homeworks because they want to see is the turtle really going to move the way I think or the way I wanted to move, okay. And basically now you have put students in a position of power and everybody loves power and they are going to want to make the turtle dance to their tune.

(Refer Slide Time: 26:05)



Okay, so now the next topic is going to be how computers solve problems. So we will discuss this. This is another part of the introductory first few lectures. We will discuss this in the next sublecture. So we will take a, take a break. Thank you.