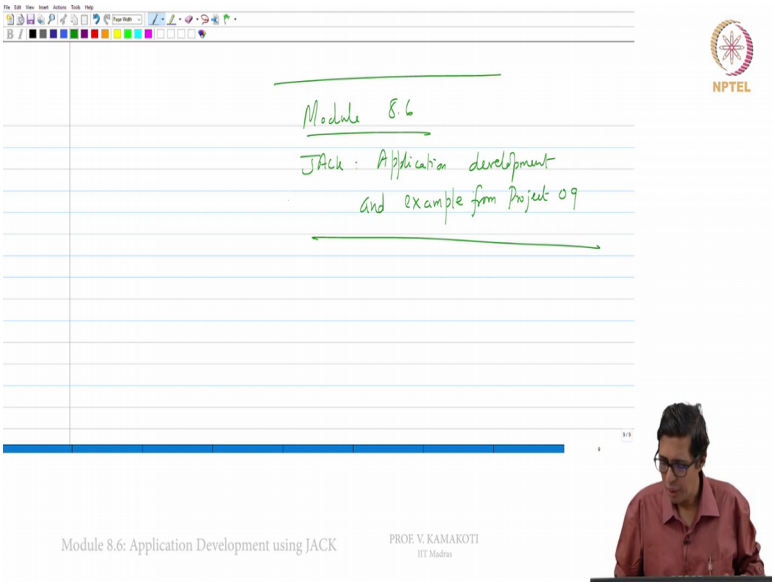
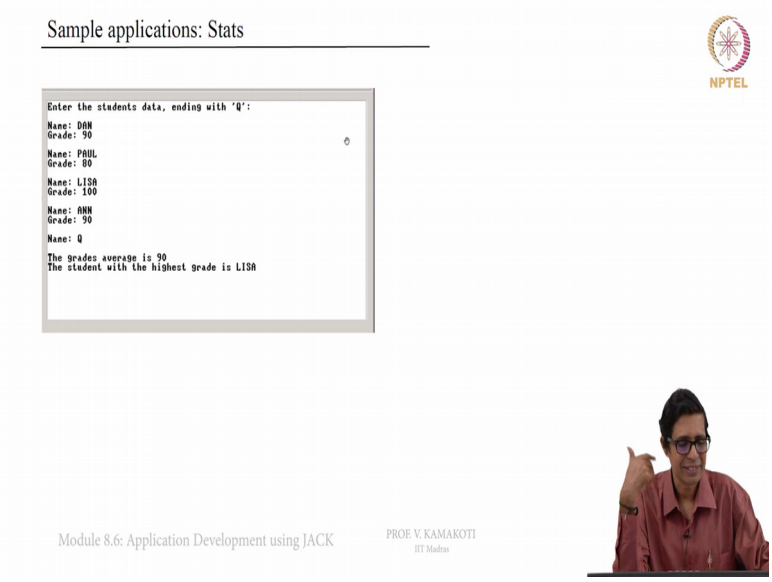
Foundations to Computer Systems Design
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Module 8.6
Application Development using Jack

(Refer Slide Time: 0:18)



So welcome to module 8.6 in which we will be covering the application development using Jack and will also use some example from project 09 to show you how this is going to work.

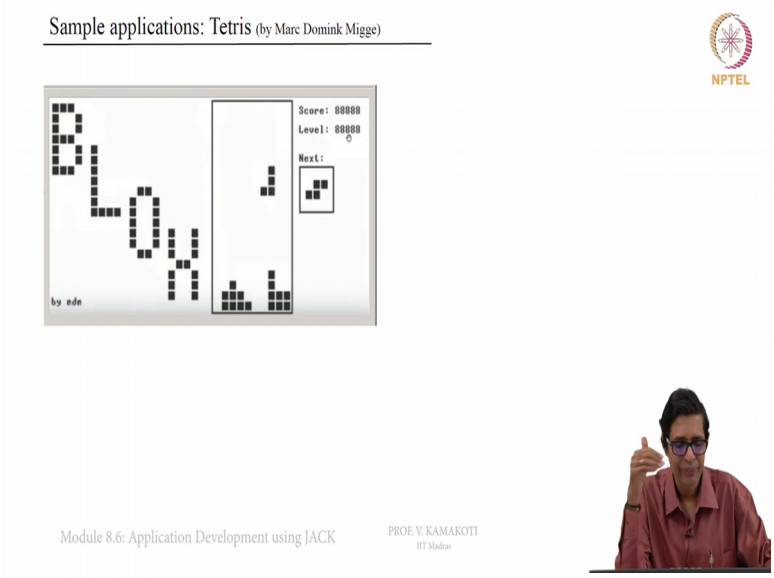
(Refer Slide Time: 0:29)

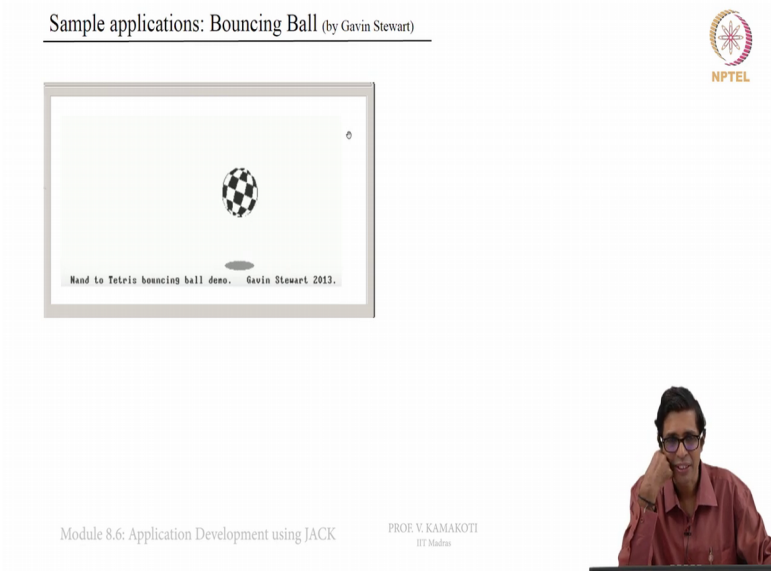


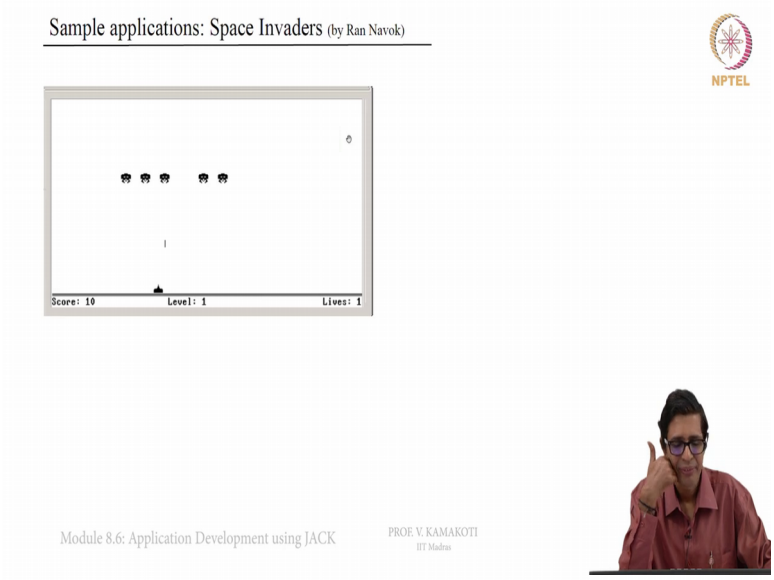
So we can write certain application like enter the student data, ending with Q right, so write a program which will except name and grade, name, grade, name, grade, name, grade,

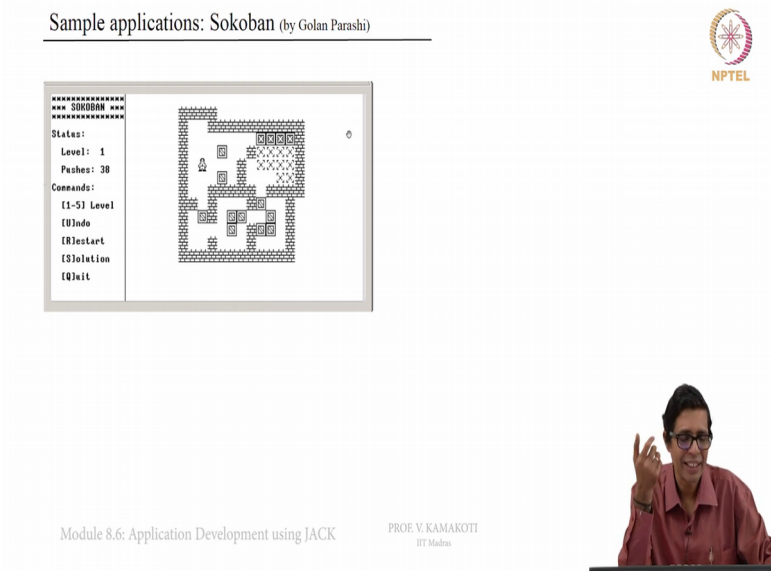
calculate, the grade, the average of their grades and also get the student with the highest grade, so we can do this, so basically we can declare an array and do this.

(Refer Slide Time: 0:52)



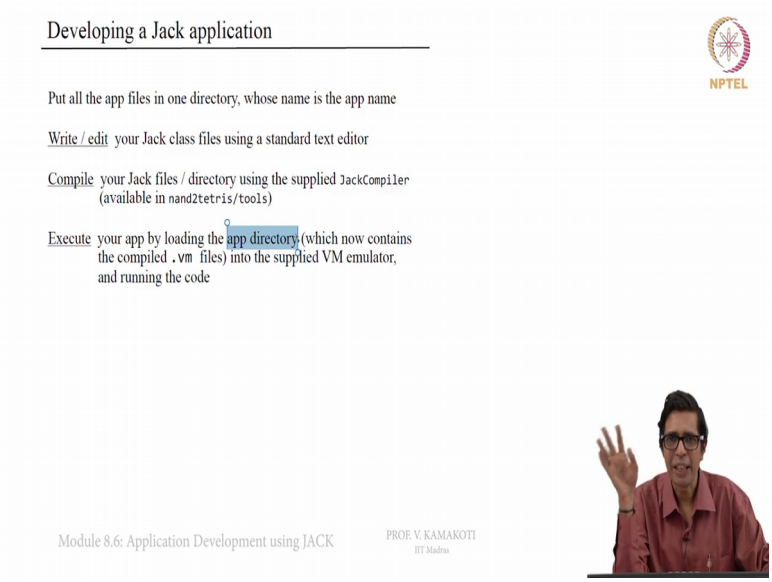






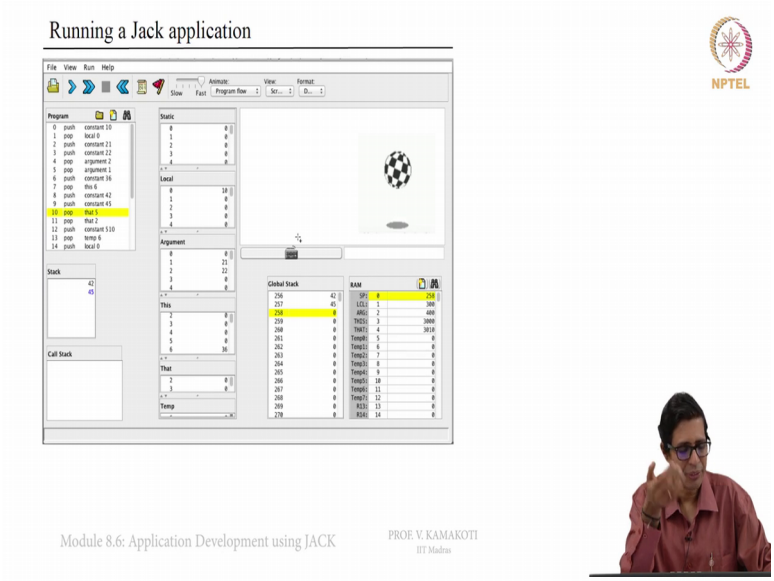
So this Tetris game which is finally we are going to play is this, this is one application that is developed, there is also an nand2tetris bouncing ball demo that has been created, these are the space invader thing, Sokoban has been done on Jack right, using Jack on hack okay.

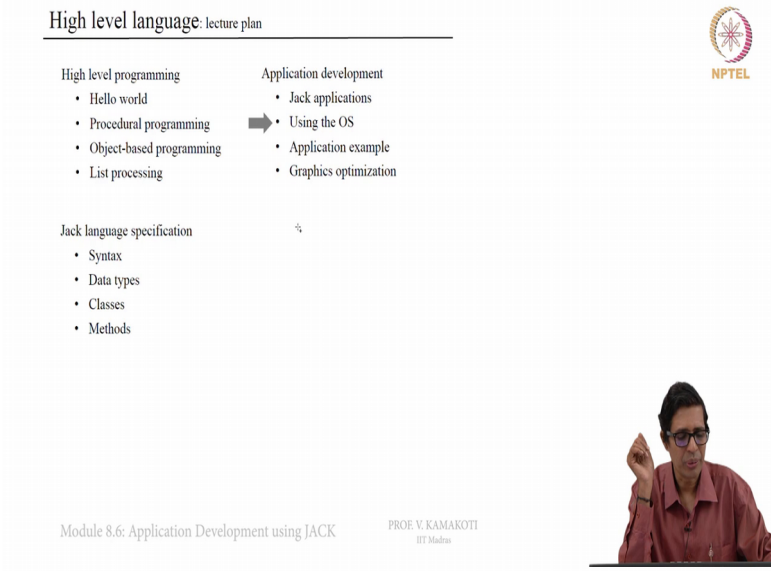
(Refer Slide Time: 1:25)

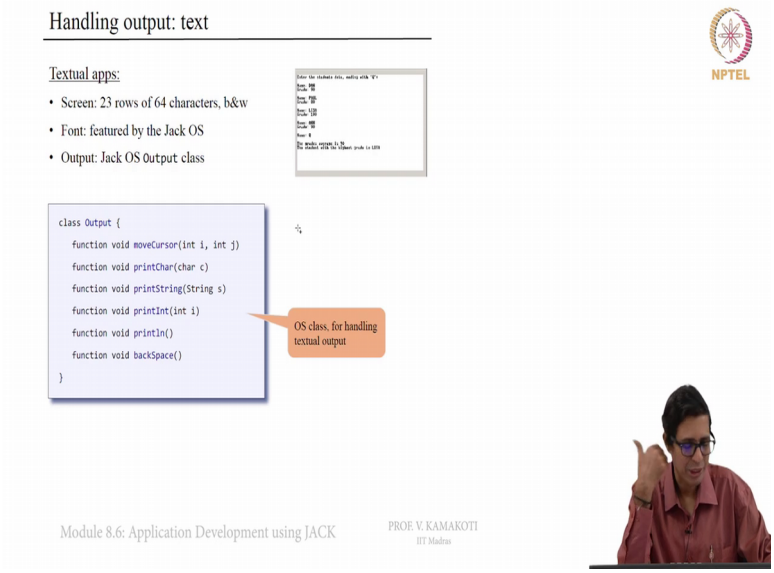


So how do you develop this whole Jack application? Put all the app files in one directory whose name is the app name.jack, write/edit your Jack class files using a standard text editor, compile your Jack files using the Jack compiler as we have shown and execute your app by loading the app directory which now contains T.vm files into the loading app directory, so if you have multiple VM files, you just click on the app directory as I showed you, on the VM emulator we can see this.

(Refer Slide Time: 2:00)

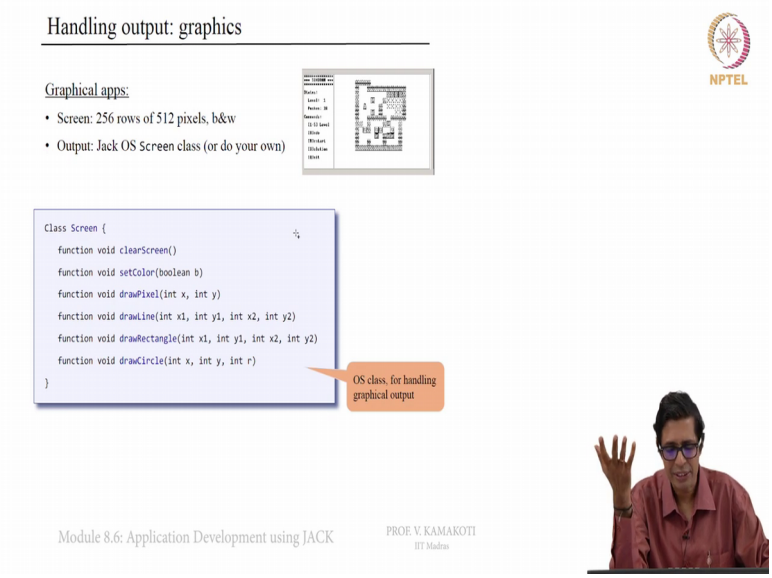






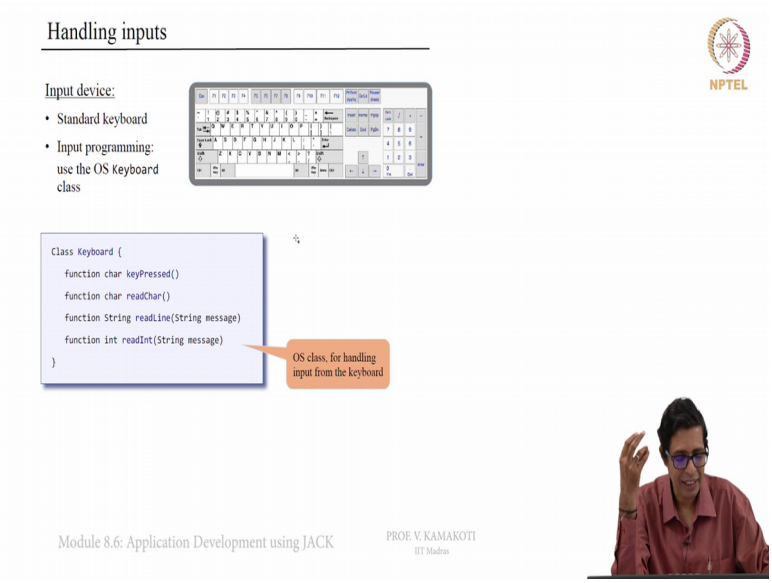
So we have seen this how to run a Jack application and using the OS we have already seen so for output, you could there are different classes in the OS, so we have class output, which can move a cursor, you can print a character, you can print string, you can print int, print LN, backspace, so all these things are possible.

(Refer Slide Time: 2:26)



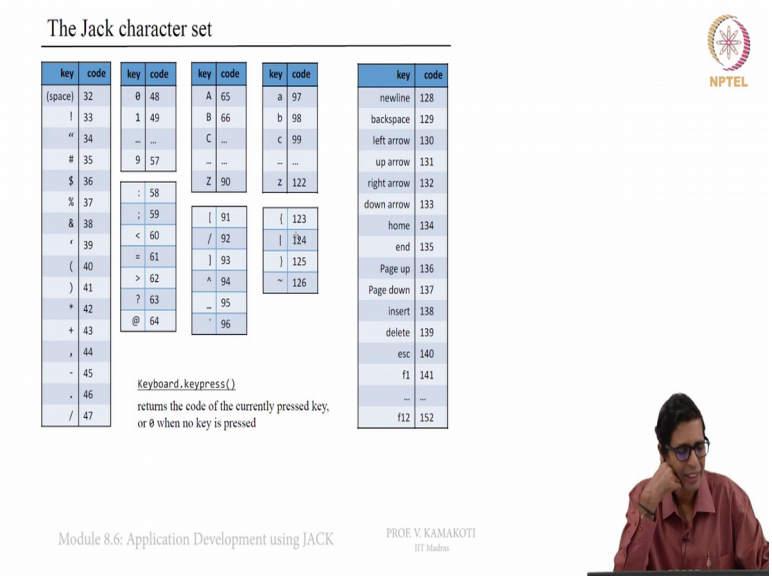
For the graphics there is a clear screen, set color, drop pixel, draw line, draw a rectangle, draw circle, so almost all these things can be done.

(Refer Slide Time: 2:33)



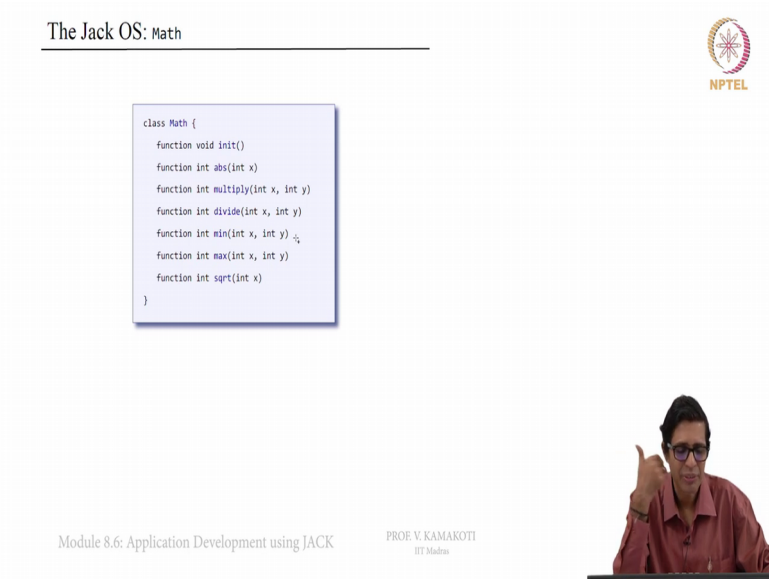
For the inputs from the keyboard I can check whether the key is pressed, I can read a character, I can read a line, I can read an int from the keyboard.

(Refer Slide Time: 2:43)



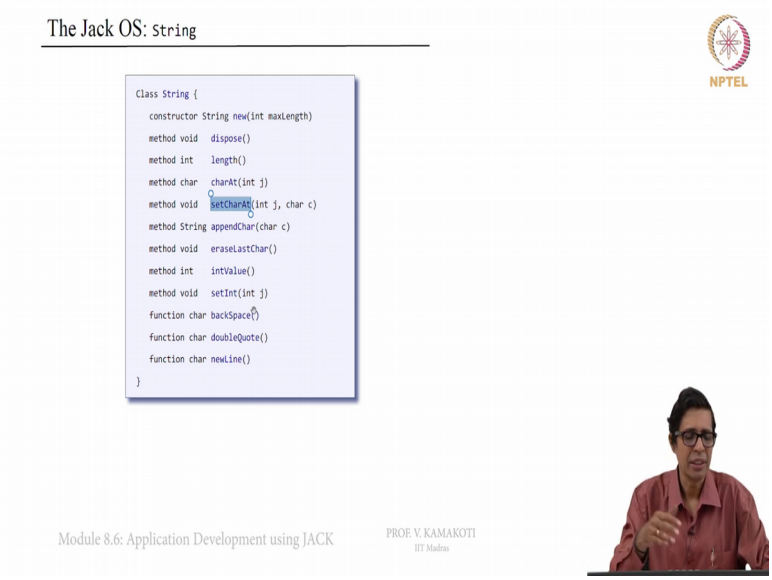
This is the entire Jack character set, so the keyboard.keypress if you use return the code of the currently pressed key or 0 when no key is pressed, so 0 is not assigned to anyone and this is the restricted Jack character set, so when I define an architecture, when I defined an OS I have to define my own character set which the hardware should interpret and give it to me.

(Refer Slide Time: 3:07)



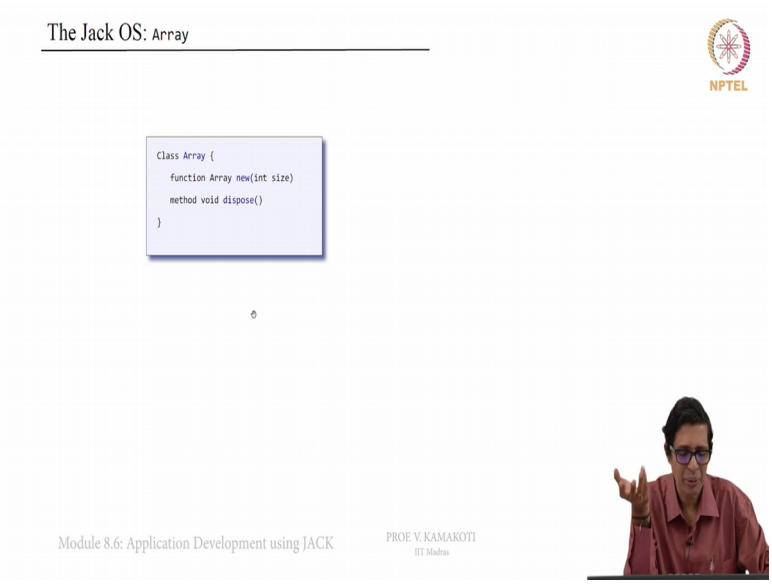
Than this is the math library, you have eight functions here, note that our hack architecture does not support multiply and divide as an hardware functionality, it only does addition, so multiplied, divide is provided as a math library here and we have main, max, square root and these absolute values and in it functions also, in it will initiate the math library.

(Refer Slide Time: 3:33)



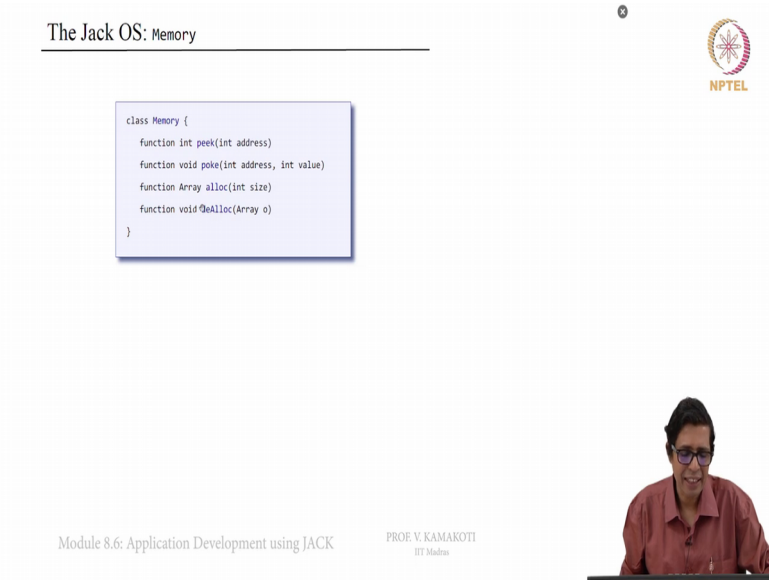
For string again a lot of things that string, new string, dispose a string, length of a string, character at particular location in the string, said the character at a location, so set char at will actually set the character C at location J, append the character at the end of the string, erase the last character, int value of the string, set int, int J, backspace, double quote, newline, so these are all of the string functionalities of the Jack OS.

(Refer Slide Time: 4:15)



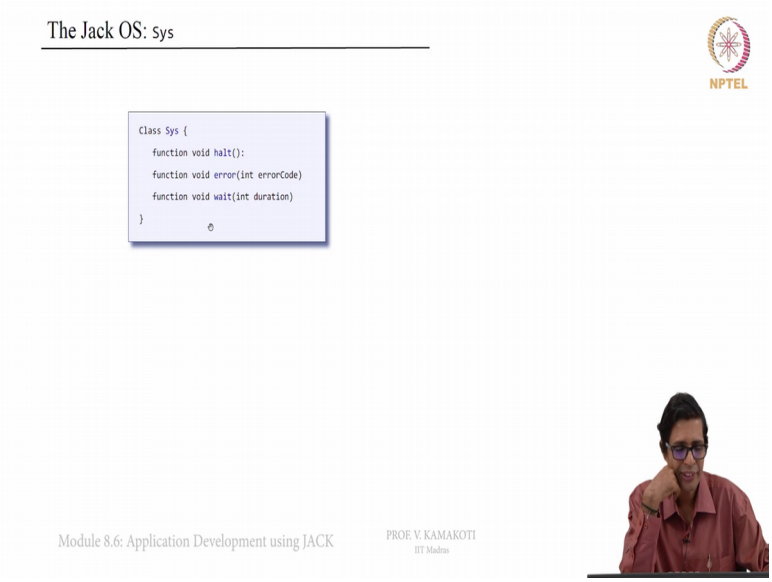
And class array, array has new and dispose and it is not type, so I can do, I can store multiple things in the same array, one location integer, one location character etc.

(Refer Slide Time: 4:25)



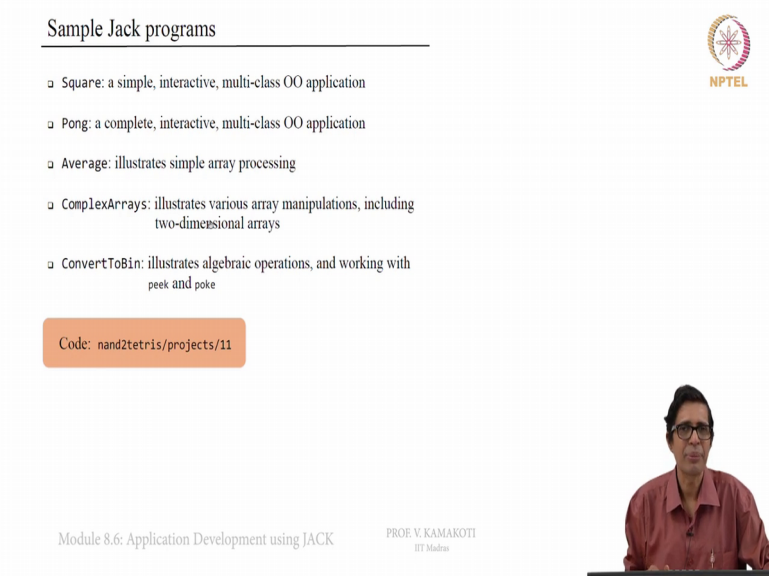
Memory, memory has the following functionalities, peak, poke, alloc, dealloc, peak means it will go and read from that address, poke means it will quote the address and write a values, so I can write into particular memory, I can allocate function array, allocate function void deallocate, I can allocate an array, so the argument is or an array memory.array, memory.alloc will give me an array of so many elements size and the allocate it takes an array and deallocates all the values there and it freezes all this things and give it back to the heap.

(Refer Slide Time: 5:11)



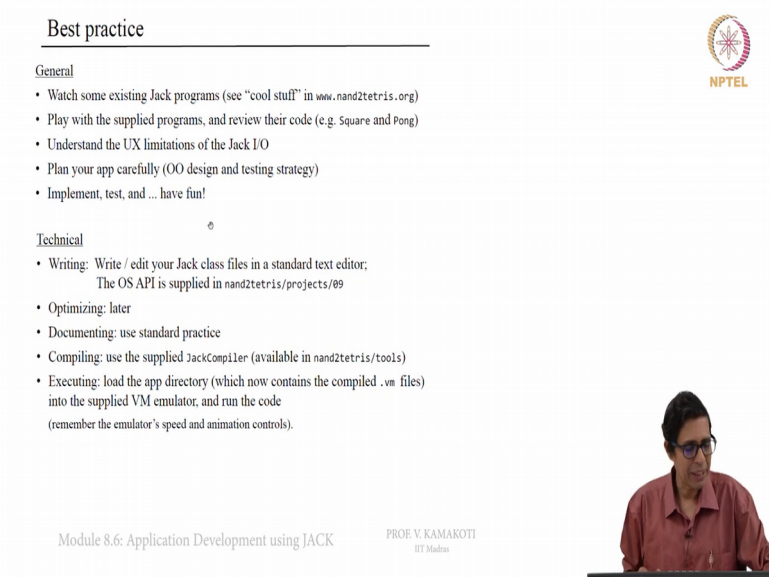
There is a sys which will halt the system, which will print an error code, which will wait for a particular duration, these are all operating systems functionalities.

(Refer Slide Time: 5:20)



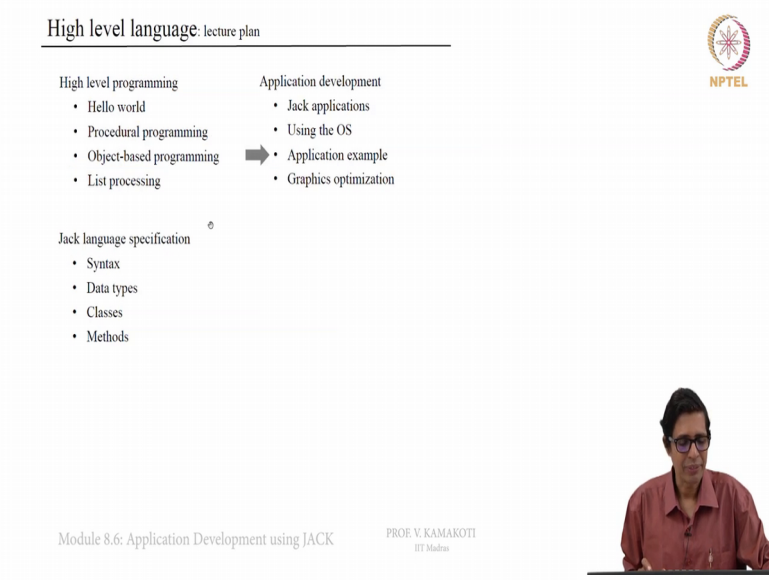
So you have seen a lot of things like square, pong, average, complex arrays, convert, so these are all sample programs that are available, you can look at this in 11th project, project 11 right, so we can see many of this Jack programs, understand this Jack programs.

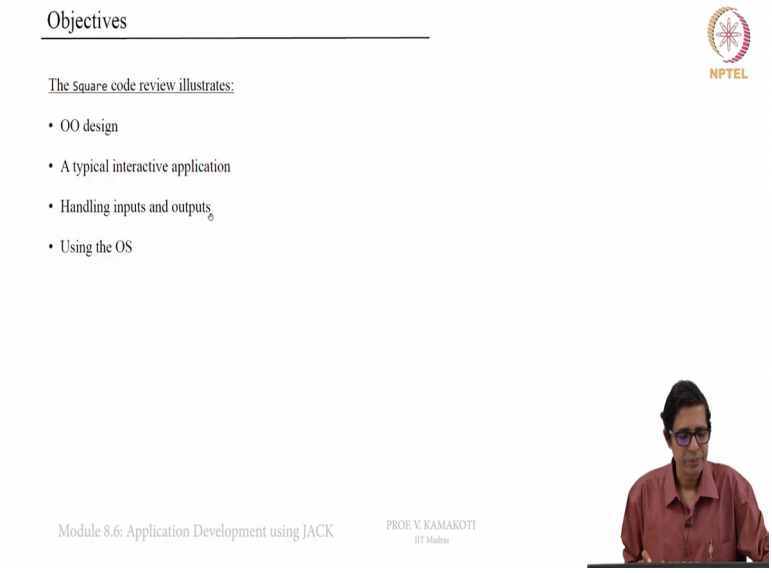
(Refer Slide Time: 5:39)

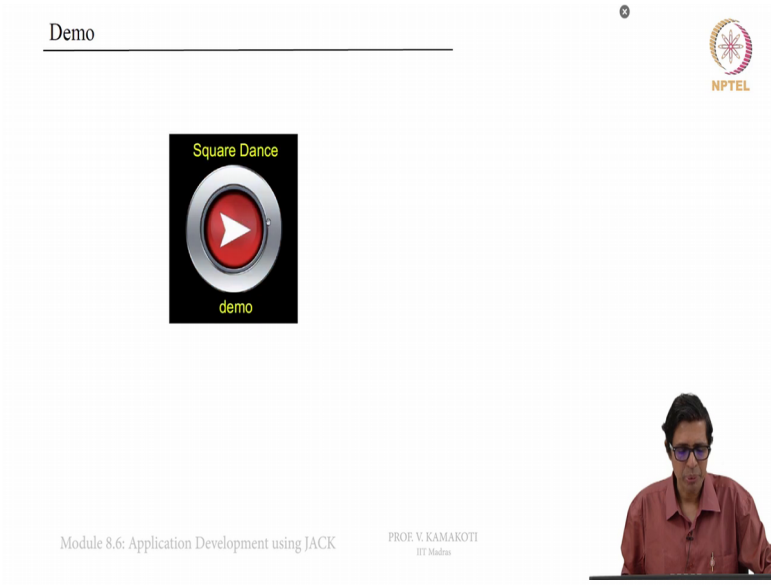


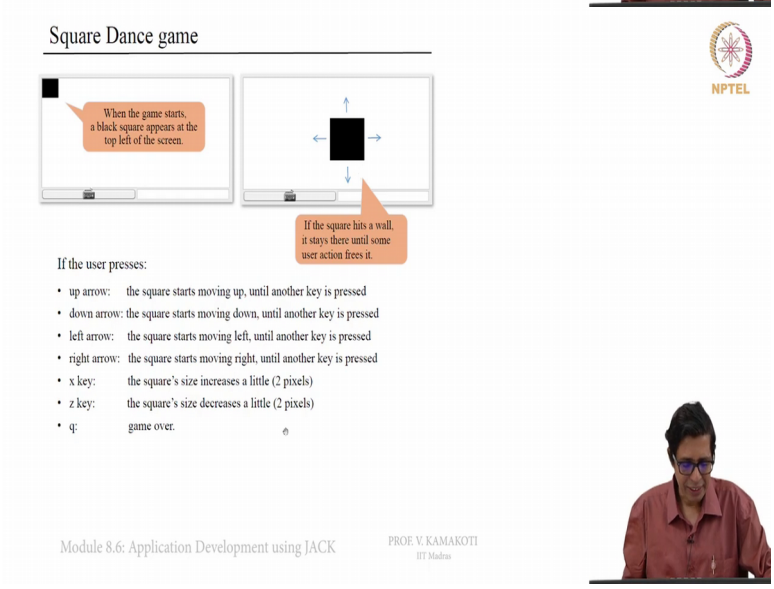
So the best practice is general watch some existing jack program in nand2tetris.org, play with the supplied programs, your square and pong, understand the you know, user interface limitations, plan your app carefully, implement, test and have fun, so the technical part is, you should write the Jack class files, optimising vision to a later, documents some of these use this as a standard practice, compile using Jack compiler, execute using the VM emulator using this. Okay.

(Refer Slide Time: 6:12)



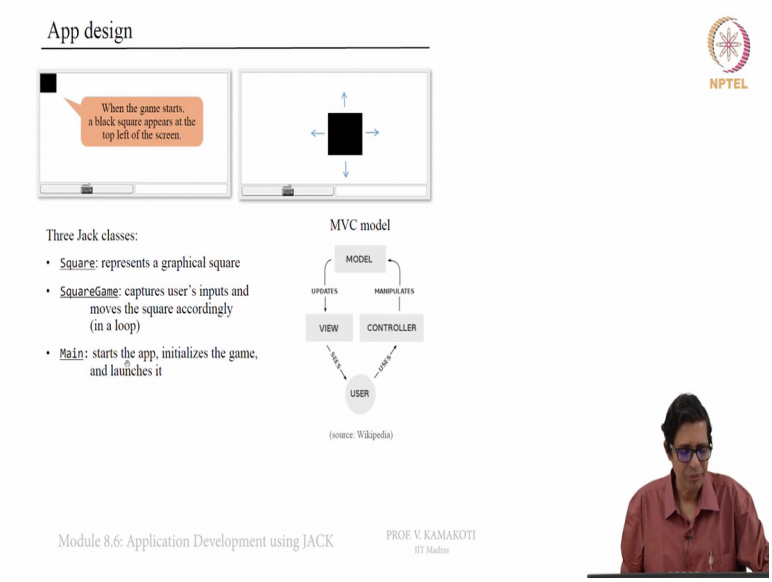






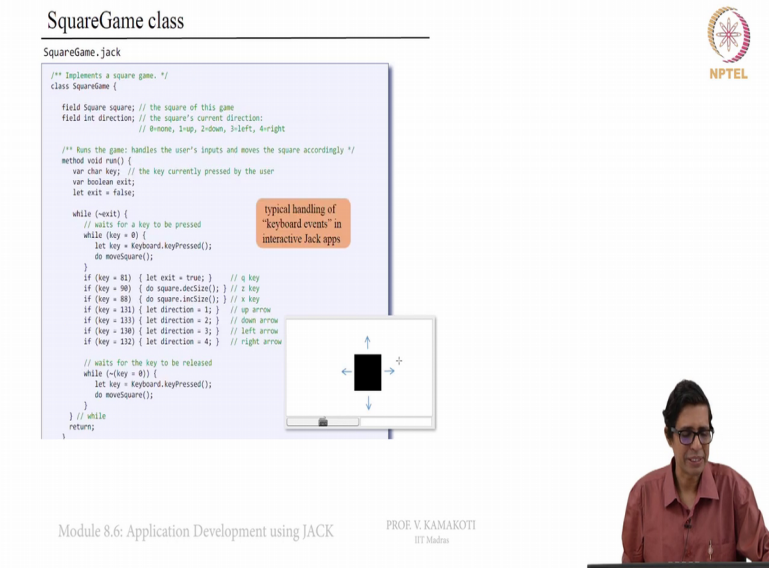
So we can see of this as a square dance demo, the square which we showed, the up arrow will take it up, down arrow will bring it down, we started looking at this, the X key is at key will decrease and increase the square size right.

(Refer Slide Time: 6:41)



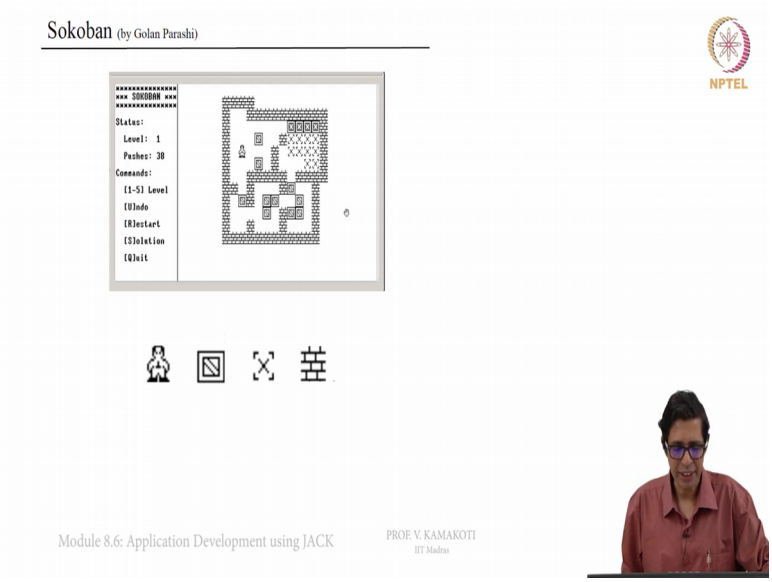
So this has three classes square, square game and main, we saw that as a part of our there exercise right.

(Refer Slide Time: 6:49)



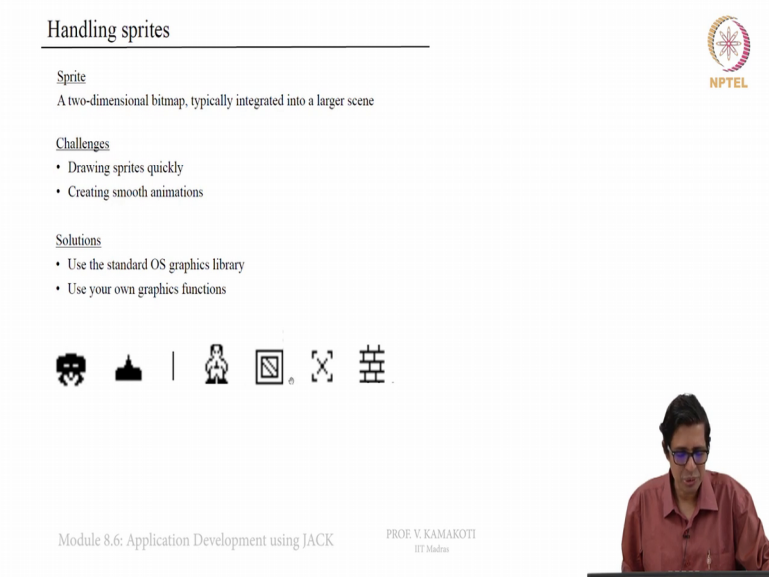
So you can look at the square, this is the functionality of the square right, so this functionalities up, down, left, right square should move and X and Z should increase or decrease in size and this Q means game over, so this is what it should do and how is this app actually designed, the set of slides basically tells you how that is designed okay.

(Refer Slide Time: 7:31)



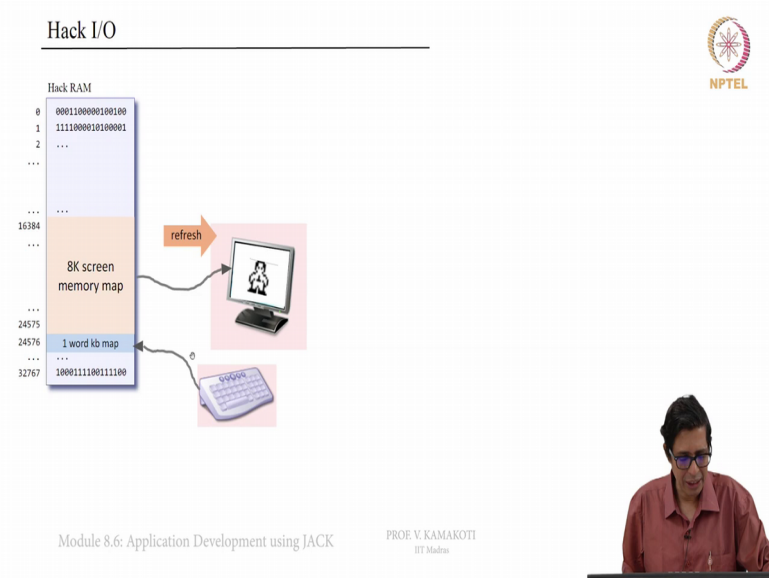
So these are nice ways because the pixels are given here, we have a limited number of pixels within that you need to create all this beautiful things graphics.

(Refer Slide Time: 7:42)



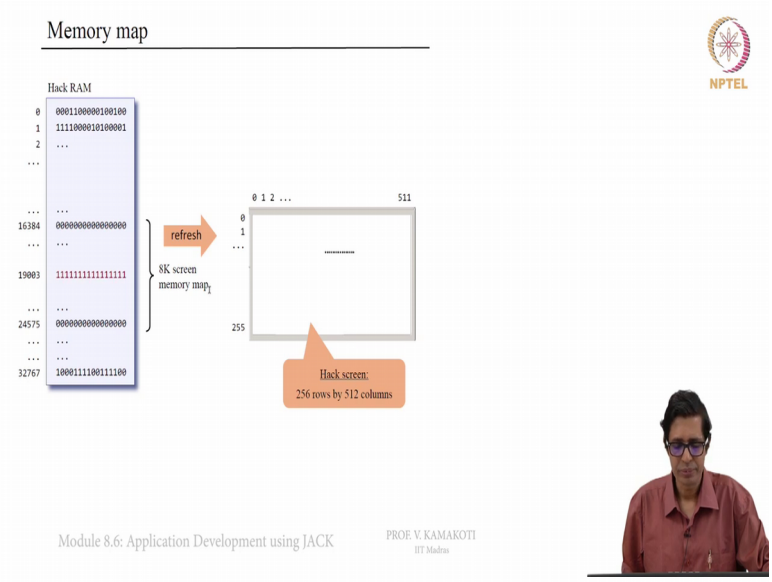
So this is all some very interesting challenges of how do you read this small graphic libraries, you can create this libraries and start using these objects one by one as a part of this.

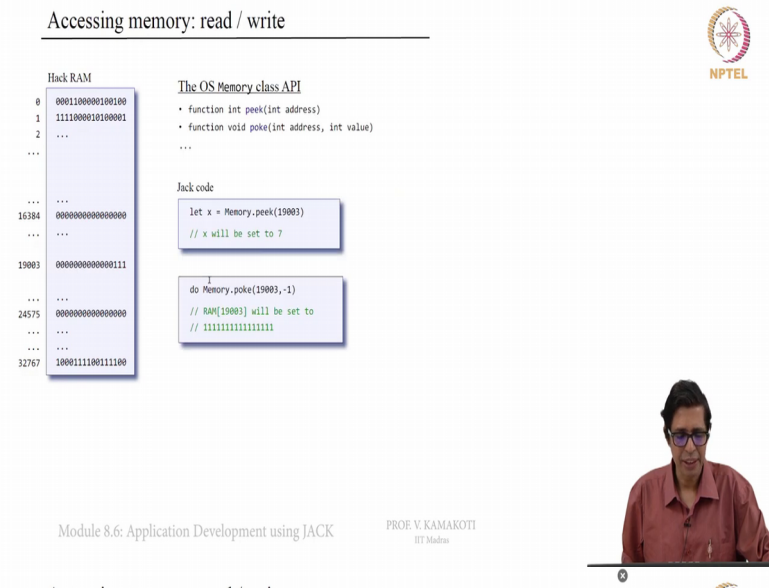
(Refer Slide Time: 7:55)

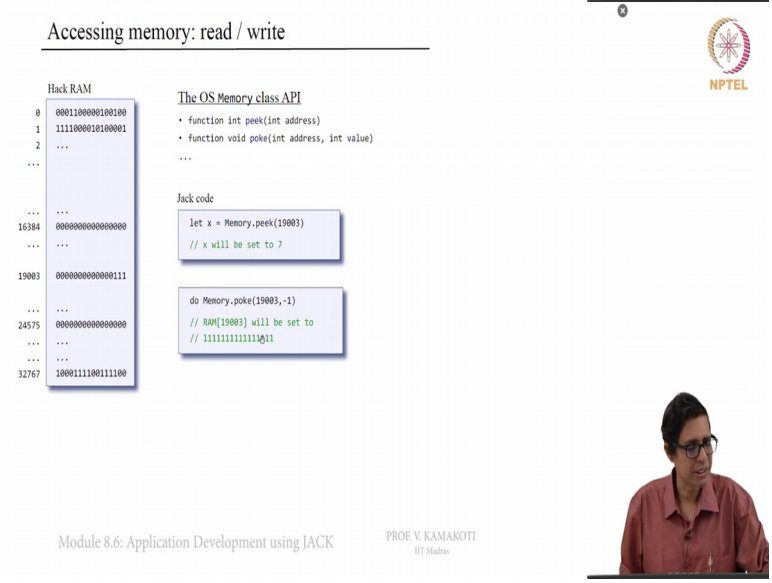


And of course the hack I/O have already seen, there is a 8 kB screen memory map and one word keyboard map right.

(Refer Slide Time: 8:04)

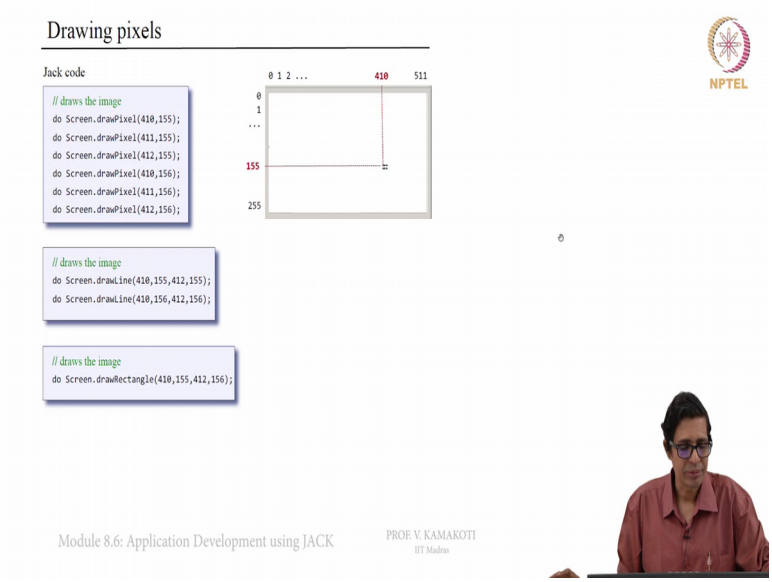






So this is how the hack, so from a Jack language can say let as is equal to memory.peak 19,003 than it will go and write, it will get a values 7 because in 19,003 actually you see the value 7 here and similarly I can say memory.poke 19,003-1 and it will go and immediately set memory to 111111, -1 in twos complement is all once right, so I can read and write memory using this.

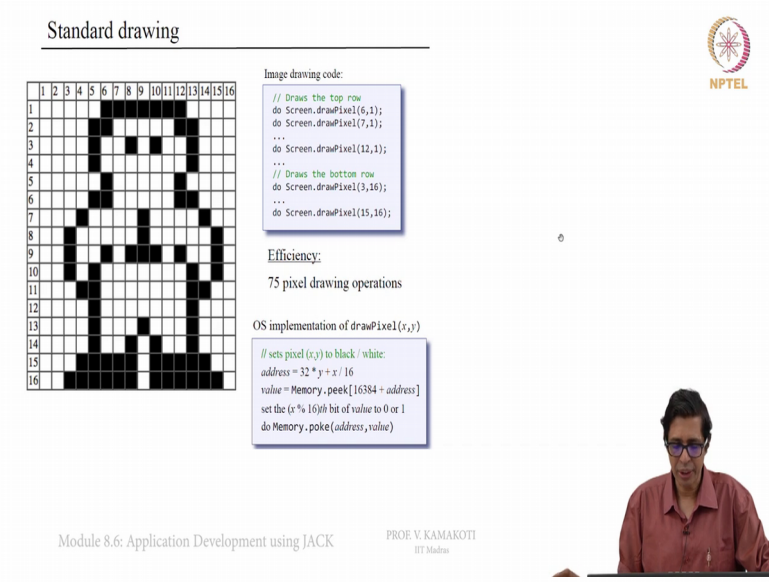
(Refer Slide Time: 8:46)

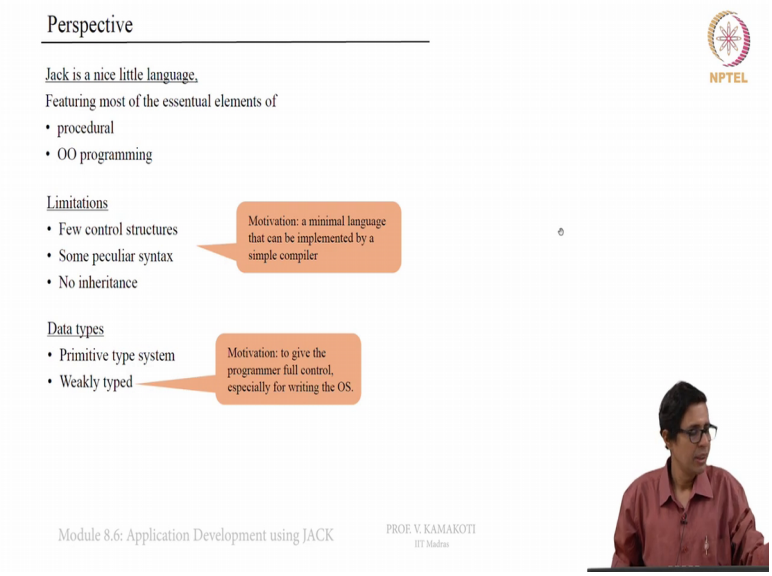


So any screen I can say draw.pixel 410, 115, so it will go and, so if I put all this screen.drawpixel it will print all this 6 dots here as you see here, so I can set the color as 1 or 0, I can draw pixel, I can draw a line right, I can draw a rectangle, I can draw a circle and these are all given as OS screen class to us right, normally this is currently done by the hack CPU in the advance computing, these functions will be done by the hardware, the graphics card right, you have seen NVIDIA and other cords, those cords will be given the responsibility of drawing this lines, etc.

So the main CPU will not do this functionalities, it will delegate it to us separate hardware but in the case of hack, the hack computer, if you say draw pixel of course there is a code that needs to execute which will draw the pixels here right, which will draw the line and rectangle, that will be coming to the hack computer and the hack computer will be executing it, but in a case of real systems, there is a graphics coprocessor, the graphics card, you are NVIDIA, test lock cord etc that will be doing on your behalf right and that we the CPUs relieved of some functionalities.

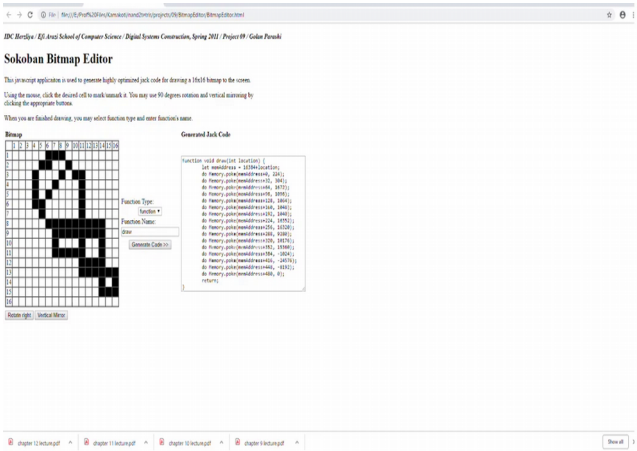
(Refer Slide Time: 10:19)





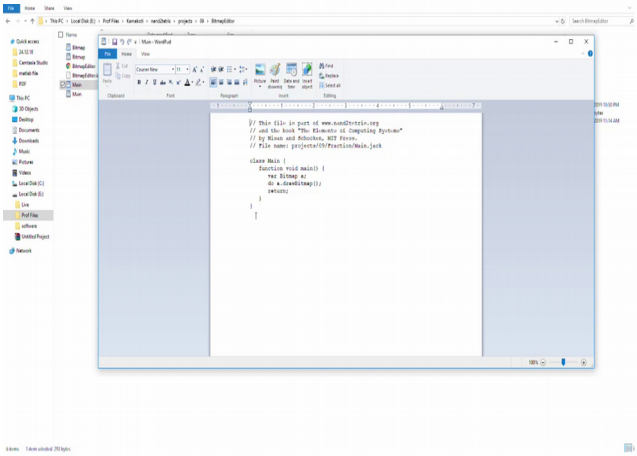
So I could have this types of, I give you, will show one demo of this type of custom drawings okay, so to sum up Jack is an very nice little language, limitation is has few control structures, some particular syntax, no inheritance type of things, it is not a fully-fledged of object-oriented type and it has very weekly typed, so this gives the grammar full control specially for writing the OS etc, so we make this language simple, enough because want to teach the essence of how to construct a compiler, the backend code, etc, and that is why this language as simple as possible right.

(Refer Slide Time: 11:15)



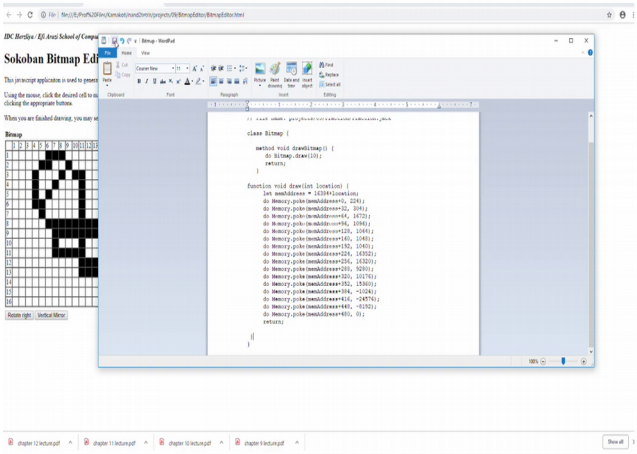
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



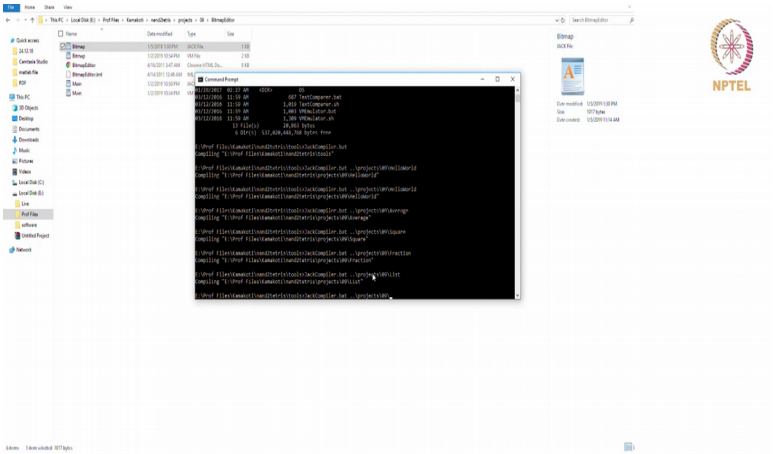
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



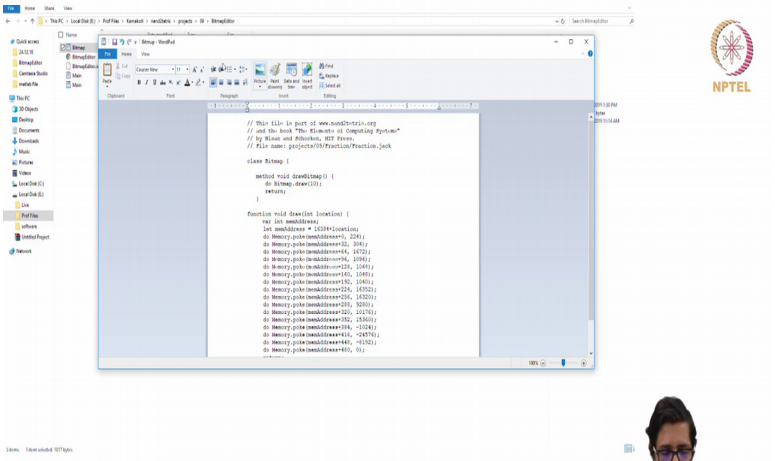
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



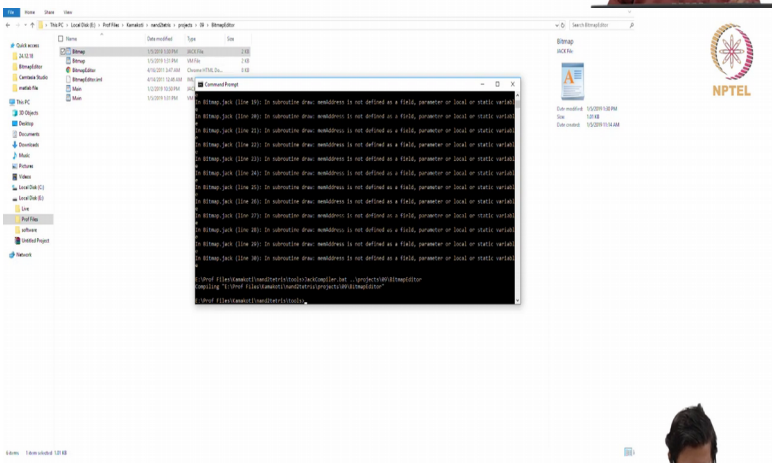
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



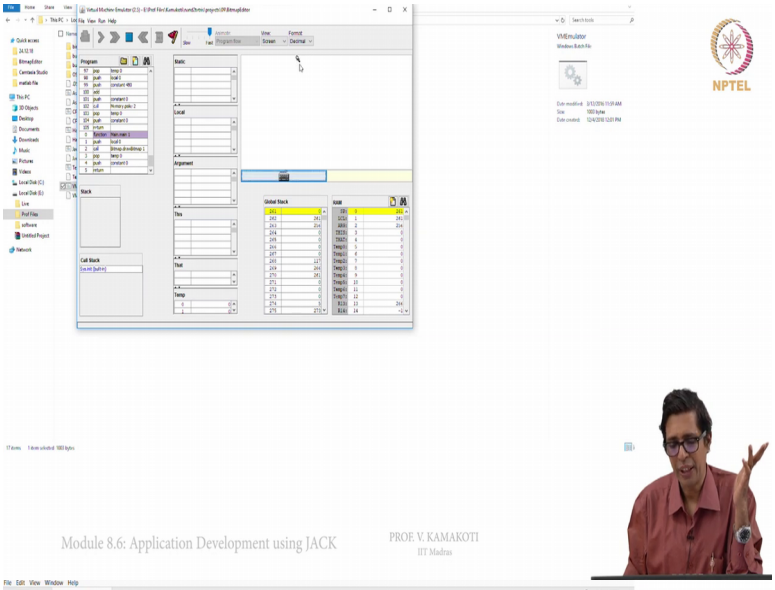
Module 8.6: Application Development using JACK

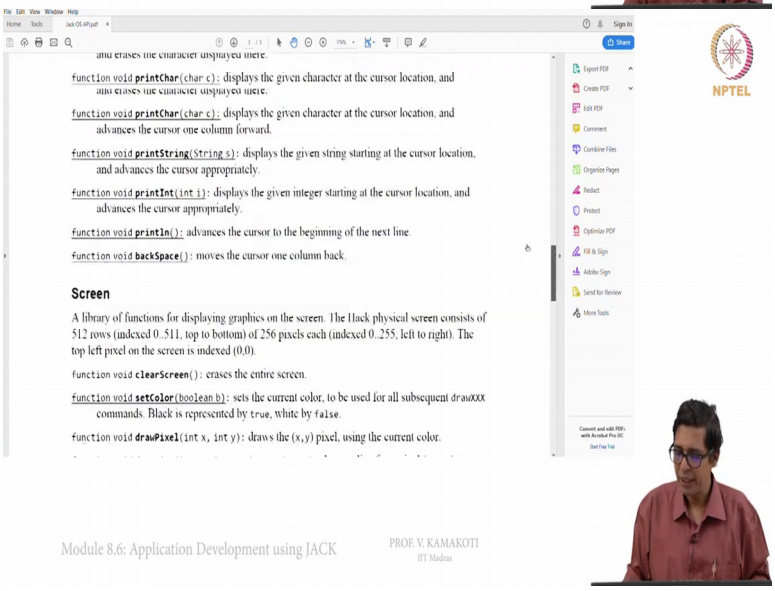
PROF. V. KAMAKOTI
IIT Madras



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras





So with this we just show very quick demo and then and this entire module, the demo is basically that on the last part basically will go and see this bitmap editor, there is a, so we can basically, so this is a completely square, Sokoban editor, so I can basically want to make some figure like this, so I can basically, so I want to make a figure like this, this is as probably as make or control, so I want to make a figure like this, so I can say generate the code and this is the code that is generated right.

Now, so this generates a Jack code, now we can go back to your thing and there is a main file, main.jack which will call this a., So this file, you need to create main.jack with class main, void main, we have var bitmap A and do a.drawbitmap and return, so bitmap is another class file that you use here, so in this bitmap you have this Jack file and then you can create this method, void, draw bitmap, do bitmap.draw10 return and then right, now from 10, so do

bitmap.draw, in this bitmap.draw whatever you have got as a part of your HTML file here this whole thing you can cut and paste here.

So you can cut and paste this entire thing, copy and then go back to this and then you can just paste it here. Okay and then right, this is the whole thing now we can save this whole file, so this is very nice interesting way of creating this applications, now we can go to your this thing and you can compile this, this is actually called as bitmap editor, such says some error is there, let us, we will just say wherein name, address and then we can basically run this code, yes, this is done now let us execute this code as part of your projects, let us go, let us take the VM emulator right and then letters load this particular program which is part of your bitmap editor, this is loaded, yes and now let us run this program at full speed and see.

Now you start seeing something happening here as a part of your yes, so this is a nice way of trying to understand the thing, so this is, so you have created a small logo here right, so this is one very nice application and then to just sum up these, if you go to your projects 09, inside 09 you have this complete PDF file of the Jack OS API, which contains whatever we have described as a part of this lecture, so enjoy doing Jack, try to do write some Jack programs, compile it using the Jack compiler, execute it on the VM machine and get familiarize with this language, at least these 5 programs, 6 programs that we have seen, we can get familiarize and there are many more programs on the net and also on the Nand2Tetris.org site, just look at those programs and that is going to be your project 09.

So this is a different project, where you do not do much, you do not program but you understand the language and basically while executing VM please go to the different steps, especially when we are calling certain operating system APIs that you have listed as a part of this right, so how does the operating system support the ecosystem of execution of the program and that is very, very important as a part of this, all the best and we will meet at module 9. Thank you.