

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)

Module 8.6
Jack: Application development
and example from Project 09

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

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)

Sample applications: Stats

```
Enter the students data, ending with 'Q':
Name: DAN
Grade: 90
Name: PAUL
Grade: 80
Name: LISA
Grade: 100
Name: ANN
Grade: 90
Name: Q
The Grades average is 90
The student with the highest grade is LISA
```

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

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)

Sample applications: Tetris (by Marc Domink Migge)



Score: 88888
Level: 88888
Next: [Next piece preview]

by rde

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

Sample applications: Bouncing Ball (by Gavin Stewart)



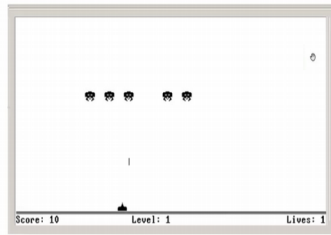
Hand to Tetris bouncing ball demo. Gavin Stewart 2013.

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



Sample applications: Space Invaders (by Ran Navok)

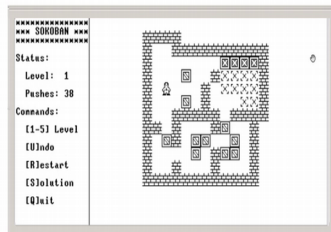


Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



Sample applications: Sokoban (by Golan Parashi)



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



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)



Developing a Jack application

Put all the app files in one directory, whose name is the app name

Write / edit your Jack class files using a standard text editor

Compile your Jack files / directory using the supplied JackCompiler (available in nand2tetris/tools)

Execute your app by loading the app directory (which now contains the compiled .vm files) into the supplied VM emulator, and running the code



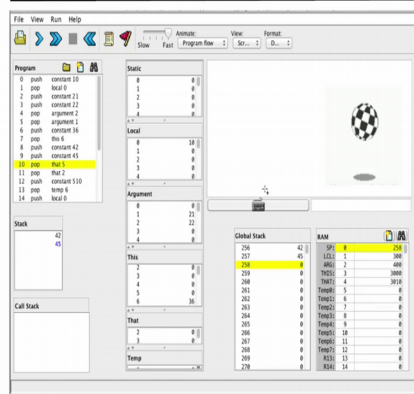
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

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)

Running a Jack application



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

High level language. lecture plan

- | | | |
|--|---|--|
| <ul style="list-style-type: none"> • High level programming • Hello world • Procedural programming • Object-based programming • List processing | ➔ | <ul style="list-style-type: none"> • Application development • Jack applications • Using the OS • Application example • Graphics optimization |
|--|---|--|

Jack language specification

- Syntax
- Data types
- Classes
- Methods



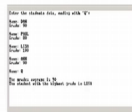
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

Handling output: text

Textual apps:

- Screen: 23 rows of 64 characters, b&w
- Font: featured by the Jack OS
- Output: Jack OS output class



```
class Output {
    function void moveCursor(int i, int j)
    function void printChar(char c)
    function void printString(String s)
    function void printInt(int i)
    function void printIn()
    function void backSpace()
}
```

OS class, for handling textual output



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

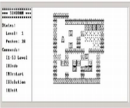
So we have seen this how to run a Jack application and using the OS we have already seen so for output, you could there are different classes in the OS, so we have class output, which can move a cursor, you can print a character, you can print string, you can print int, print LN, backspace, so all these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

(Refer Slide Time: 2:26)

Handling output: graphics

Graphical apps:

- Screen: 256 rows of 512 pixels, b&w
- Output: Jack OS Screen class (or do your own)




```
Class Screen {
    function void clearScreen()
    function void setColor(boolean b)
    function void drawPixel(int x, int y)
    function void drawLine(int x1, int y1, int x2, int y2)
    function void drawRectangle(int x1, int y1, int x2, int y2)
    function void drawCircle(int x, int y, int r)
}
```

OS class, for handling graphical output

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras




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)

Handling inputs

Input device:

- Standard keyboard
- Input programming: use the OS Keyboard class




```
Class Keyboard {
    function char keyPressed()
    function char readChar()
    function String readLine(String message)
    function int readInt(String message)
}
```

OS class, for handling input from the keyboard

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



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)

The Jack character set

key	code	key	code	key	code	key	code
(space)	32	0	48	A	65	a	97
!	33	1	49	B	66	b	98
"	34	--	--	C	--	c	99
#	35	9	57	--	--	--	--
\$	36	:	58	Z	90	z	122
%	37	;	59	[91	{	123
&	38	<	60	/	92		124
'	39	=	61]	93	}	125
(40	>	62	^	94	~	126
)	41	?	63	--	--	--	--
*	42	@	64	--	--	--	--
+	43						
,	44						
-	45						
.	46						
/	47						

`Keyboard.keypress()`
returns the code of the currently pressed key,
or 0 when no key is pressed



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)

The Jack OS: Math

```
class Math {
  function void init()
  function int abs(int x)
  function int multiply(int x, int y)
  function int divide(int x, int y)
  function int min(int x, int y) ;
  function int max(int x, int y)
  function int sqrt(int x)
}
```



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)

The Jack OS: String

```
Class String {
  constructor String new(int maxlength)
  method void dispose()
  method int length()
  method char charAt(int j)
  method void setCharAt(int j, char c)
  method String appendChar(char c)
  method void eraseLastChar()
  method int intValue()
  method void setInt(int j)
  function char backspace()
  function char doubleQuote()
  function char newline()
}
```

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



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)

The Jack OS: Array

```
Class Array {
  function Array new(int size)
  method void dispose()
}
```

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras




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)


The Jack OS: Memory

```
class Memory {  
    function int peek(int address)  
    function void poke(int address, int value)  
    function Array alloc(int size)  
    function void DeAlloc(Array o)  
}
```



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



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)


The Jack OS: sys

```
Class Sys {  
    function void halt():  
    function void error(int errorCode)  
    function void wait(int duration)  
}
```



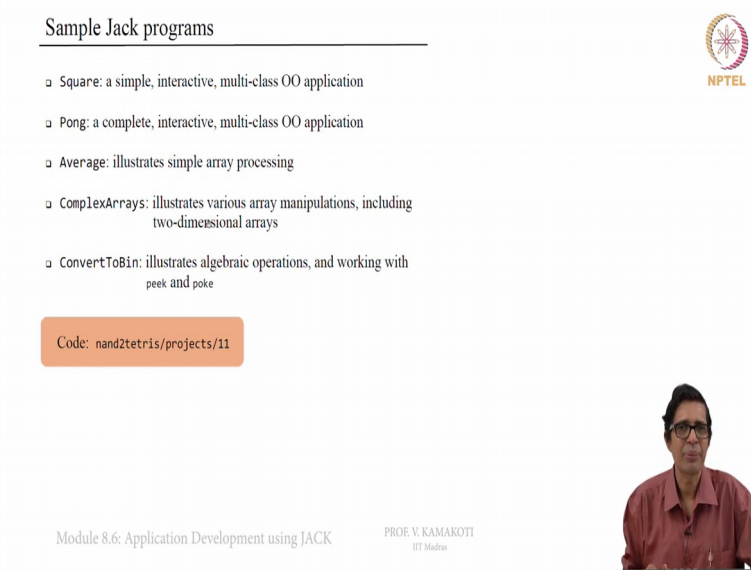
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



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)





Sample Jack programs

- Square: a simple, interactive, multi-class OO application
- Pong: a complete, interactive, multi-class OO application
- Average: illustrates simple array processing
- ComplexArrays: illustrates various array manipulations, including two-dimensional arrays
- ConvertToBin: illustrates algebraic operations, and working with peek and poke

Code: `nand2tetris/projects/11`

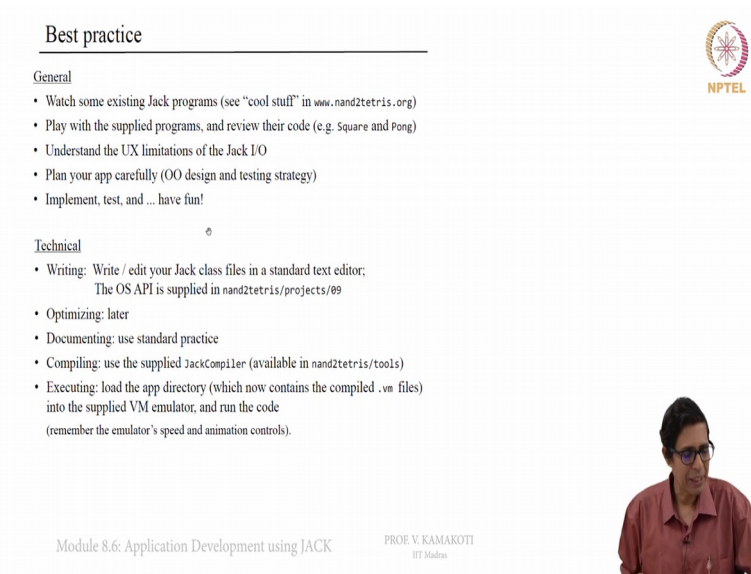
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



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)



Best practice

General



- Watch some existing Jack programs (see “cool stuff” in www.nand2tetris.org)
- Play with the supplied programs, and review their code (e.g. Square and Pong)
- Understand the UX limitations of the Jack I/O
- Plan your app carefully (OO design and testing strategy)
- Implement, test, and ... have fun!

Technical

- Writing: Write / edit your Jack class files in a standard text editor.
The OS API is supplied in `nand2tetris/projects/09`
- Optimizing: later
- Documenting: use standard practice
- Compiling: use the supplied `JackCompiler` (available in `nand2tetris/tools`)
- Executing: load the app directory (which now contains the compiled `.vm` files) into the supplied VM emulator, and run the code (remember the emulator's speed and animation controls).

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras



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)

High level language: lecture plan

High level programming	Application development
<ul style="list-style-type: none">• Hello world• Procedural programming• Object-based programming• List processing	<ul style="list-style-type: none">• Jack applications• Using the OS• Application example• Graphics optimization

Jack language specification

- Syntax
- Data types
- Classes
- Methods

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras





Objectives

The Square code review illustrates:

- OO design
- A typical interactive application
- Handling inputs and outputs
- Using the OS

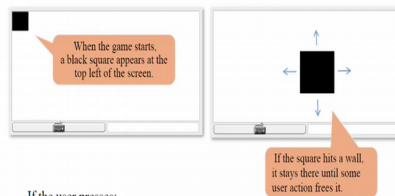
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras





Square Dance game



If the user presses:


- up arrow: the square starts moving up, until another key is pressed
- down arrow: the square starts moving down, until another key is pressed
- left arrow: the square starts moving left, until another key is pressed
- right arrow: the square starts moving right, until another key is pressed
- x key: the square's size increases a little (2 pixels)
- z key: the square's size decreases a little (2 pixels)
- q: game over.



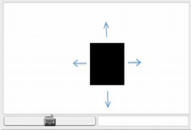
So we can see of this as a square dance demo, the square which we showed, the up arrow will take it up, down arrow will bring it down, we started looking at this, the X key is at key will decrease and increase the square size right.

(Refer Slide Time: 6:41)

App design




When the game starts, a black square appears at the top left of the screen.



Three Jack classes:

- **Square**: represents a graphical square
- **SquareGame**: captures user's inputs and moves the square accordingly (in a loop)
- **Main**: starts the app, initializes the game, and launches it

MVC model




```
graph TD;
    Model[MODEL] -- UPDATES --> View[VIEW];
    View -- DATA --> Controller[CONTROLLER];
    Controller -- MANIPULATES --> Model;
    User((USER)) -- INPUTS --> Controller;
```

(source: Wikipedia)

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



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)

Square class API

```
/** Implements a graphical square. */
class Square {
    /** Constructs a new square with a given location and size */
    constructor Square new(int Ax, int Ay, int ASize)

    /** Disposes this square */
    method void dispose()

    /** Draws this square on the screen */
    method void draw()

    /** Erases this square from the screen */
    method void erase()

    /** Increments this square's size by 2 pixels */
    method void incSize()

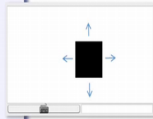
    /** Decrements this square's size by 2 pixels */
    method void decSize()

    /** Moves this square up by 2 pixels */
    method void moveUp()

    /** Moves this square down by 2 pixels */
    method void moveDown()

    /** Moves this square left by 2 pixels */
    method void moveLeft()

    /** Moves this square right by 2 pixels */
    method void moveRight()
}
```



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



SquareGame class

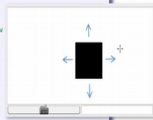
```
/** Implements a square game. */
class SquareGame {
    field Square square; // the square of this game
    field int direction; // the square's current direction:
                        // 0:none, 1:up, 2:down, 3:left, 4:right

    /** Runs the game: handles the user's inputs and moves the square accordingly */
    method void run() {
        var char key; // the key currently pressed by the user
        var boolean exits;
        let exits = false;

        while (~exits) {
            // waits for a key to be pressed
            while (key = 0) {
                let key = Keyboard.keyPressed();
                do moveSquare();
            }
            if (key = 81) { let exits = true; } // q key
            if (key = 90) { do square.decSize(); } // z key
            if (key = 88) { do square.incSize(); } // x key
            if (key = 113) { let direction = 1; } // up arrow
            if (key = 133) { let direction = 2; } // down arrow
            if (key = 138) { let direction = 3; } // left arrow
            if (key = 132) { let direction = 4; } // right arrow

            // waits for the key to be released
            while (~key = 0) {
                let key = Keyboard.keyPressed();
                do moveSquare();
            }
        } // while
        return;
    }
}
```

typical handling of
"keyboard events" in
interactive Jack apps



Module 8.6: Application Development using JACK


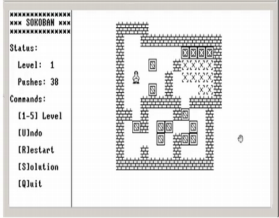
PROF. V. KAMAKOTI
IIT Madras



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)

Sokoban (by Golan Parashi)



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



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)

Handling sprites



Sprite
A two-dimensional bitmap, typically integrated into a larger scene

Challenges

- Drawing sprites quickly
- Creating smooth animations


Solutions

- Use the standard OS graphics library
- Use your own graphics functions



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



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)

Hack I/O

The diagram illustrates the memory layout of the Hack I/O system. It features a vertical stack of memory regions:

- Hack RAM:** A large blue box at the top containing binary data (0001100000100100, 1111000010100001, ...).
- 8K screen memory map:** An orange box in the middle, labeled "8K screen memory map", with an arrow pointing to a computer monitor icon. An orange arrow labeled "refresh" also points to the monitor.
- 1 word kb map:** A small blue box at the bottom, labeled "1 word kb map", with an arrow pointing to a keyboard icon.

Memory addresses are listed on the left side of the RAM and screen map regions: 0, 1, 2, ..., 16384, ..., 24575, 24576, ..., 32767.

NPTEL

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

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)

Memory map

NPTEL

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

Accessing memory: read / write

```
Hack RAM
0 0001100000100100
1 1111000010100001
2 ...
...
16384 0000000000000000
...
19003 0000000000000111
...
24575 0000000000000000
...
32767 1000111100111100
```

The OS Memory class API

- function int peek(int address)
- function void poke(int address, int value)
- ...

Jack code

```
let x = Memory.peek(19003)
// x will be set to 7
```

```
do Memory.poke(19003,-1)
// RAM[19003] will be set to
// 1111111111111111
```

NPTEL

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

Accessing memory: read / write

```
Hack RAM
0 0001100000100100
1 1111000010100001
2 ...
...
16384 0000000000000000
...
19003 0000000000000111
...
24575 0000000000000000
...
32767 1000111100111100
```

The OS Memory class API

- function int peek(int address)
- function void poke(int address, int value)
- ...

Jack code

```
let x = Memory.peek(19003)
// x will be set to 7
```

```
do Memory.poke(19003,-1)
// RAM[19003] will be set to
// 1111111111111111
```

NPTEL

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

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)

Drawing pixels

Jack code

```
// draws the image
do Screen.drawPixel(410,155);
do Screen.drawPixel(411,155);
do Screen.drawPixel(412,155);
do Screen.drawPixel(410,156);
do Screen.drawPixel(411,156);
do Screen.drawPixel(412,156);
```

```
// draws the image
do Screen.drawLine(410,155,412,155);
do Screen.drawLine(410,156,412,156);
```

```
// draws the image
do Screen.drawRect(410,155,412,156);
```

0 1 2 ... 410 511

0
1
...
155
255

NPTEL

Module 8.6: Application Development using JACK

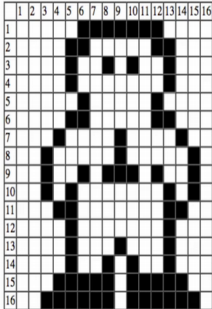
PROF. V. KAMAKOTTI
IIT Madras

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)

Standard drawing



```
Image drawing code:  
// Draws the top row  
do Screen.drawPixel(6,1);  
do Screen.drawPixel(7,1);  
...  
do Screen.drawPixel(12,1);  
...  
// Draws the bottom row  
do Screen.drawPixel(3,16);  
...  
do Screen.drawPixel(15,16);
```

Efficiency:
75 pixel drawing operations

OS implementation of drawPixel(x,y)

```
// sets pixel (x,y) to black / white  
address = 32 * y + x / 16  
value = Memory.peek[16384 + address]  
set the (x % 16)th bit of value to 0 or 1  
do Memory.poke(address, value)
```

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras

Perspective

Jack is a nice little language,
Featuring most of the essential elements of

- procedural
- OO programming

Limitations

- Few control structures
- Some peculiar syntax
- No inheritance

Motivation: a minimal language that can be implemented by a simple compiler

Data types

- Primitive type system
- Weakly typed

Motivation: to give the programmer full control, especially for writing the OS.

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTTI
IIT Madras

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)

Sokoban Bitmap Editor

This JavaScript application is used to generate highly optimized Jack code for drawing a 16x16 bitmap to the screen. Using the mouse, click the desired cell to mark it black. You may use 90 degree rotation and vertical mirroring by clicking the appropriate buttons.

When you are finished drawing, you may select function type and name function's name.



Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

```

// This file is part of www.kamakoti.org
// and the book "The Elements of Computer Systems"
// by Niklas and Achilleas, MIT Press
// File name: project9/9/FunctionMain.jack

class Main {
    function void main() {
        let bitmap as
        do x.drawBitmap()
    }
}
    
```



Lines: 1 (selected) 20 lines

Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras

```

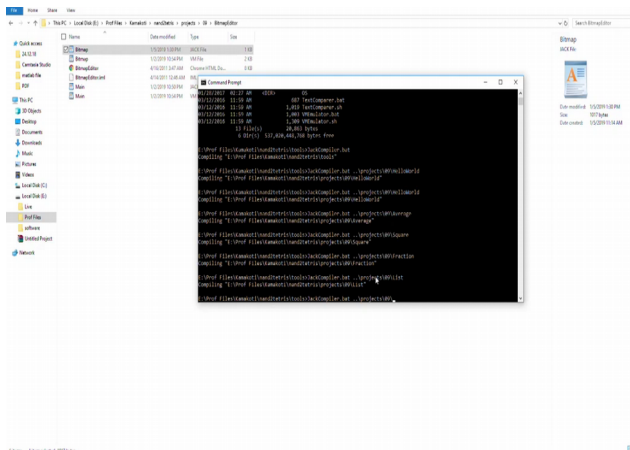
class Bitmap {
    method void drawBitmap() {
        do bitmap.draw(16)
        return
    }

    function void drawCell (rowIndex) {
        let cellIndex as 1234567890123456
        do memory.poke(memoryAddress+0, 224)
        do memory.poke(memoryAddress+1, 204)
        do memory.poke(memoryAddress+2, 204)
        do memory.poke(memoryAddress+3, 247)
        do memory.poke(memoryAddress+4, 204)
        do memory.poke(memoryAddress+5, 160)
        do memory.poke(memoryAddress+6, 160)
        do memory.poke(memoryAddress+7, 160)
        do memory.poke(memoryAddress+8, 160)
        do memory.poke(memoryAddress+9, 160)
        do memory.poke(memoryAddress+10, 160)
        do memory.poke(memoryAddress+11, 160)
        do memory.poke(memoryAddress+12, 160)
        do memory.poke(memoryAddress+13, 160)
        do memory.poke(memoryAddress+14, 160)
        do memory.poke(memoryAddress+15, 160)
        do memory.poke(memoryAddress+16, 160)
        do memory.poke(memoryAddress+17, 160)
        do memory.poke(memoryAddress+18, 160)
        do memory.poke(memoryAddress+19, 160)
        do memory.poke(memoryAddress+20, 160)
        do memory.poke(memoryAddress+21, 160)
        do memory.poke(memoryAddress+22, 160)
        do memory.poke(memoryAddress+23, 160)
        do memory.poke(memoryAddress+24, 160)
        do memory.poke(memoryAddress+25, 160)
        do memory.poke(memoryAddress+26, 160)
        do memory.poke(memoryAddress+27, 160)
        do memory.poke(memoryAddress+28, 160)
        do memory.poke(memoryAddress+29, 160)
        do memory.poke(memoryAddress+30, 160)
        do memory.poke(memoryAddress+31, 160)
        do memory.poke(memoryAddress+32, 160)
        do memory.poke(memoryAddress+33, 160)
        do memory.poke(memoryAddress+34, 160)
        do memory.poke(memoryAddress+35, 160)
        do memory.poke(memoryAddress+36, 160)
        do memory.poke(memoryAddress+37, 160)
        do memory.poke(memoryAddress+38, 160)
        do memory.poke(memoryAddress+39, 160)
        do memory.poke(memoryAddress+40, 160)
        return
    }
}
    
```



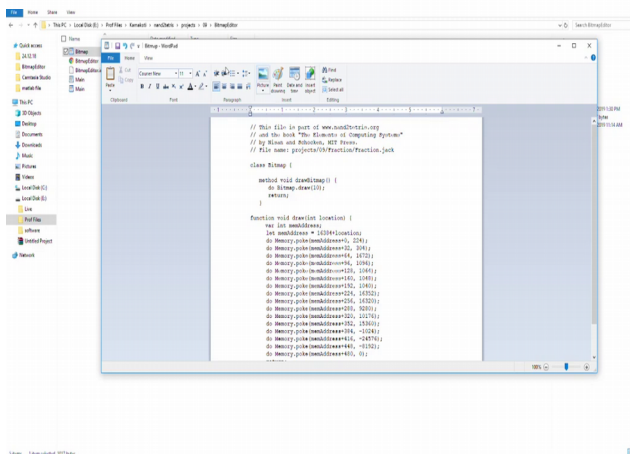
Module 8.6: Application Development using JACK

PROF. V. KAMAKOTI
IIT Madras



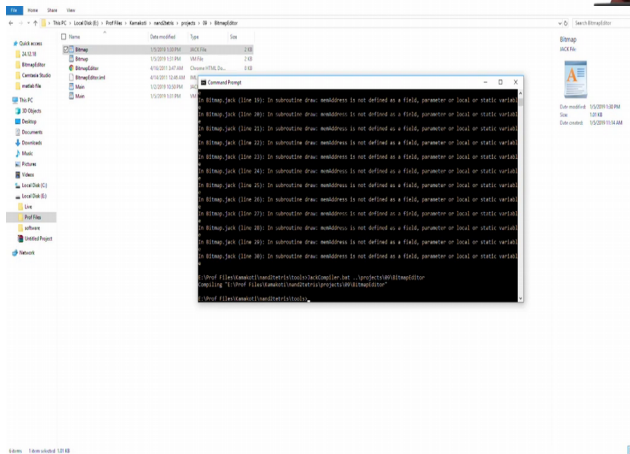
Module 8.6: Application Development using JACK

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.