// Quantum Circuit Visualizer

// Vivek Shende, 7 July 2001

// Instructions:

// have any other applet/script call the public method

// loadFile(fileName) wher fileName is a url specified

// relative to the calling document.

// File Format:

// this visualizer reads files

// in which the first line is of the format

// numQubits numQGates

// and every following line is of the format

// gateType onQubits controlQubits gateData

// where gateType is a character, onQubits specifies which qubits

// operates on, controlQubits specifies the control qubits,

// and gateData is additional data dependent on the gate type

// all decomposition programs used here output in this format.

import java.applet.Applet;

import java.awt.\*;

import java.io.\*;

import java.util.StringTokenizer;

import java.net.URL;

public class QCircuitDisplay extends Applet {

// begin smooth scrolling code

// taken directly out of an article posted at

// http://www.webresource.net/java/articles/buffering/

// thanx to Alex Pelaez

Image offscreenImage;

Graphics offg;

public void update(Graphics g) {

Dimension d = getSize();

// Create the offscreen buffer if necessary.

if (offscreenImage == null ||

offscreenImage.getWidth(null) != d.width ||

offscreenImage.getHeight(null) != d.height) {

// first time, or after resize

offscreenImage = createImage(d.width, d.height);

if (offg != null)

offg.dispose();

offg = offscreenImage.getGraphics();

}

// Clear the buffer to the applet's background color.

offg.setColor(getBackground());

offg.fillRect(0, 0, d.width, d.height);

offg.setColor(getForeground());

// Call paint with the offscreen buffer.

paint(offg);

// Draw the buffer to the screen.

g.drawImage(offscreenImage, 0, 0, this);

}

public void destroy() {

if (offg != null)

offg.dispose();

}

// end smooth scrolling code

final int kCircuitTop = 0;

final int kCircuitLeft = 30;

final int kCircuitBot = 0;

final int kCircuitRight = 30;

final int kQubitDist = 25;

final int kQGateDist = 25;

final int kNotOffset = 4;

final int kCntRadius = 4;

final int kBoxOffset = 10;

final int kTextOffsetV = 4;

final int kTextOffsetH = 4;

int numDispGates;

String QCSpec[];

int numQubits;

int numQGates;

int startGate;

public void loadFile(String filename){

try{

BufferedReader in=new BufferedReader(new InputStreamReader

((new URL(getDocumentBase(),filename)).openStream()));

String line;

StringTokenizer token;

if ((line=in.readLine())!=null)

;

token=new StringTokenizer(line," ");

numQubits=(Integer.valueOf(token.nextToken())).intValue();

numQGates=(Integer.valueOf(token.nextToken())).intValue();

QCSpec=new String[numQGates];

int i;

for(i=0;(line=in.readLine())!=null;i++)

QCSpec[i]=line;

startGate=0;

}

catch(IOException e)

{

}

}

public void init(){

loadFile("opt\_cyb-3.qc");

setBackground(Color.white);

}

int xc(int qgate)

{return kCircuitLeft+kQGateDist\*(qgate-startGate)+20;}

int yc(int qubit)

{return kCircuitTop+kQubitDist\*qubit;}

void paintNot(Graphics g, int qubit, int qgate){

g.setColor(Color.black);

g.drawLine(xc(qgate)-kNotOffset,

yc(qubit)-kNotOffset,

xc(qgate)+kNotOffset,

yc(qubit)+kNotOffset);

g.drawLine(xc(qgate)-kNotOffset,

yc(qubit)+kNotOffset,

xc(qgate)+kNotOffset,

yc(qubit)-kNotOffset);

}

void paintCtrl(Graphics g, int cqubit, int qubit, int qgate){

g.setColor(Color.black);

g.drawLine(xc(qgate),yc(cqubit),xc(qgate),yc(qubit));

g.fillOval(xc(qgate)-kCntRadius,yc(cqubit)-kCntRadius,

2\*kCntRadius, 2\*kCntRadius);

}

void paintBox(Graphics g, int qubit, int qgate, char c){

g.setColor(Color.black);

g.fillRect(xc(qgate)-kBoxOffset, yc(qubit)-kBoxOffset,

2\*kBoxOffset, 2\*kBoxOffset);

g.setColor(Color.white);

g.fillRect(xc(qgate)-kBoxOffset+1, yc(qubit)-kBoxOffset+1,

2\*kBoxOffset-2, 2\*kBoxOffset-2);

g.setColor(Color.black);

g.drawString(new Character(c).toString(),

xc(qgate)-kTextOffsetH,

yc(qubit)+kTextOffsetV);

}

void paintQubit(Graphics g, int qubit){

g.setColor(Color.black);

g.drawLine(kCircuitLeft,yc(qubit),

getSize().width-1-kCircuitRight,yc(qubit));

g.drawString(new Integer(qubit).toString(),

20-kTextOffsetH, yc(qubit)+kTextOffsetV);

}

void paintArrows(Graphics g)

{

int c=getSize().height/2;

int r=getSize().width-1;

int lArrowY[] = {c,c+5,c-5};

int lArrowX[] = {3,7,7};

int rArrowY[] = {c,c-5,c+5};

int rArrowX[] = {r-3,r-7,r-7};

g.fillPolygon(lArrowX,lArrowY,3);

g.fillPolygon(rArrowX,rArrowY,3);

g.drawLine(10,0,10,getSize().height-1);

g.drawLine(r-10,0,r-10,getSize().height-1);

}

int first\_bit(int x)

{

if (x==0) return 0;

int b;

for(b=0;((x>>b) & 1)==0;b++);

return b+1;

}

public void paint(Graphics g){

numDispGates=(getSize().width-1-kCircuitLeft-kCircuitRight)/

kQGateDist;

g.setColor(Color.black);

g.drawRect(0,0,getSize().width-1,getSize().height-1);

paintArrows(g);

int q;

for(q=1;q<=numQubits;q++) paintQubit(g,q);

StringTokenizer token;

int i;

for(i=startGate;(i<numQGates) &&

(i<startGate+numDispGates);i++){

token = new StringTokenizer(QCSpec[i]," ");

char gateType=token.nextToken().charAt(0);

int o = first\_bit((Integer.valueOf(token.nextToken())).intValue());

int c = first\_bit((Integer.valueOf(token.nextToken())).intValue());

if (c!=0) paintCtrl(g,c,o,i);

switch(gateType){

case 'n':paintNot(g,o,i); break;

default:

{

double d=(Double.valueOf(token.nextToken())).doubleValue();

paintBox(g,o,i,Character.toUpperCase(gateType));

} break;

}

}

}

public boolean scroll(int x)

{

if ((startGate+x>0) && (startGate+x+numDispGates<numQGates))

{

startGate+=x;

repaint();

return true;

}

return false;

}

public boolean scroll(String x)

{return scroll((Integer.valueOf(x)).intValue());}

public boolean mouseDown(Event e, int x, int y ) {

if (x<10) scroll(-1);

else if (x>getSize().width-1-10) scroll (1);

return true;

}

}