รายการอ้างอิง

ภาษาไทย

เบนซ์ อุดตมธนินทร์ และพีรพล สุระพงษ์ชัย. Macromedia Flash 5 in Classroom. กรุงเทพมหานคร: บริษัท เอช. เอ็น. กรุ๊ป จำกัด, 2544.

ปุณณรัตน์ พิชญไพบูลย์. Computer graphics สำหรับนักออกแบบ. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2542.

มงคล อัศวโกวิทกรณ์. การเขียนโปรแกรมกราฟิก. กรุงเทพมหานคร: บริษัทดวงกมลสมัยจำกัด, 2534.

วุฒิพงศ์ พงศ์สุวรรณ และโมไนย เมตกรุณ์จิตต์. Macromedia Flash 5 Advanced Programming. พระนครศรีอยุธยา: บริษัท ซอฟต์แวร์ปาร์ค จำกัด, 2544.

ภาษาอังกฤษ

CATT-Acoutic V 8.0 demo Manual. Gothenburg Sweden: 2002.

Duncan Templet. Acoustic in the built environment advice for the design team. 1 st ed. Oxford: An imprint of Butterworth-Heinemann Ltd, 1993.

Erick Veva, Wayne Wieselev and others. Macromedia flash MX help. Sanfrancisco: Macromedia,Inc, 2002.

Flash exchanges. Available from: http://www.macromedia.com/cfusion/exchange/index.html [2003,April 14].

Flash simulation site. Available from: http://www.flash sim.com/flash-resources-flash.html [2003,April 14].

Flash support center. Available from: http://www.macromedia.com/support/flash/index.html [2003,April 22].

Harrington, S. Computer graphics a programing approach. 2 nd ed. Sigapore: McGraw-Hill Book Co., 1987.

Henry J.Cowan. Handbook of architectural technology. New York: Van Nostrand Rein Hold, 1991.

M. David Egan. Concepts in architectural acoustic. New York: McGraw-Hill Book Company, 1972.

Michael Barron. Auditorium acoustics and architectural design.1 st ed. London: E&FN Spon and imprint of Chapman & Hall, 1993.

ภาคผนวก

ภาคผนวก ก.
ตารางค่าสัมประสิทธิ์การดูดซับเสียงของวัสดุต่าง ๆ

	Sound Absorption Coefficient						NRC
Material	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz	Number*
Walls(1, 2, 5)							
Sound-reflecting:							
1. Brick, unglazed	0.03	0.03	0.03	0.04	0.05	0.07	0.05
2. Brick, unglazed and painted	0.01	0.01	0.02	0.02	0.02	0.03	0.00
3. Concrete block, painted	0.10	0.05	0.06	0.07	0.09	0.08	0.05
4. Cork on brick or concrete	0.02	0.03	0.03	0.03	0.03	0.02	0.05
5. Glass, heavy plate	0.18	0.06	0.04	0.03	0.02	0.02	0.05
6. Glass, typical window	0.35	0.25	0.18	0.12	0.07	0.04	0.15
7. Gypsum board, 1/2-in. paneling	0.29	0.10	0.05	0.04	0.07	0.09	0.05
8. Marble or glazed tile	0.01	0.01	0.01	0.01	0.02	0.02	0.00
9. Metal venetian blinds	0.06	0.05	0.07	0.15	0.13	0.17	0.10
10. Plaster, gypsum or lime, on brick	0.01	0.02	0.02	0.03	0.04	0.05	0.05
11. Plaster, gypsum or lime, on concrete block	0.12	0.09	0.07	0.05	0.05	0.04	0.05
12. Plaster, gypsum or lime, on lath	0.14	0.10	0.06	0.05	0.04	0.03	0.05
13. Plywood, 3/8-in. paneling	0.28	0.22	0.17	0.09	0.10	0.11	0.15
14. Wood, 1/4-in. paneling, with air space behind	0.42	0.21	0.10	0.08	0.06	0.06	0.10
Sound-absorbing:							
15. Concrete block, coarse	0.36	0.44	0.31	0.29	0.39	0.25	0.35
16. Cork, 1 in. with air space behind	0.14	0.25	0.40	0.25	0.34	0.21	0.30
17. Lightweight drapery, 10 oz/sq yd, flat on wall (note: sound-reflecting at most frequencies)	0.03	0.04	0.11	0.17	0.24	0.35	0.15
18. Mediumweight drapery, 14 oz/sq yd, draped to half area	0.07	0.31	0.49	0.75	0.70	0.60	0.55
19. Heavyweight drapery, 18 oz/sq yd, draped to half area	0.14	0.35	0.55	0.72	0.70	0.65	0.60
20. Fiberglas fabric curtain, 8 1/2 oz/sq yd, draped to half area	0.09	0.32	0.68	0.83	0.39	0.76	0.55
21. Shredded wood fiberboard, 2 in. thick on concrete (mounting No. 4)	0.32	0.37	0.77	0.99	0.79	0.88	0.75
22. Thick, porous sound-absorbing material with open facing	0.60	0.75	0.82	0.80	0.60	0.38	0.75
23. Carpet, heavy, on 5/8 in. perforated mineral fiber-board with air space behind	0.37	0.41	0.63	0.85	0.96	0.92	0.70

Material	Sound Absorption Coefficient						NRC
	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz	Number*
24. Wood, 1/2-in. paneling, perforated 3/16-in.-diameter holes, 11% open area, with 2 1/2-in. glass fiber in air space behind	0.40	0.90	0.80	0.50	0.40	0.30	0.65
Floors(5)							
Sound-reflecting:							
25. Concrete or terrazzo	0.01	0.01	0.02	0.02	0.02	0.02	0.00
26. Cork, rubber, linoleum, or asphalt tile on concrete	0.02	0.03	0.03	0.03	0.03	0.02	0.05
27. Marble or glazed tile	0.01	0.01	0.01	0.01	0.02	0.02	0.00
28. Wood	0.15	0.11	0.10	0.07	0.06	0.07	0.10
29. Wood parquet on concrete	0.04	0.04	0.07	0.06	0.06	0.07	0.05
Sound-absorbing:							
30. Carpet, heavy, on concrete	0.02	0.06	0.14	0.37	0.60	0.65	0.30
31. Carpet, heavy, on foam rubber	0.08	0.24	0.57	0.69	0.71	0.73	0.55
32. Carpet, heavy, with impermeable latex backing on foam rubber	0.08	0.27	0.39	0.34	0.48	0.63	0.35
33. Indoor-outdoor carpet	0.01	0.05	0.10	0.20	0.45	0.65	0.20
Ceilings(5)							
Sound-reflecting:							
34. Concrete	0.01	0.01	0.02	0.02	0.02	0.02	0.00
35. Gypsum board, 1/2-in. thick	0.29	0.10	0.05	0.04	0.07	0.09	0.05
36. Plaster, gypsum or lime, on lath	0.14	0.10	0.06	0.05	0.04	0.03	0.05
37. Plywood, 3/8-in. thick	0.28	0.22	0.17	0.09	0.10	0.11	0.15
Sound-absorbing:†							
38. Suspended acoustical tile, 3/4-in. thick (mounting No. 7)	0.76	0.93	0.83	0.99	0.99	0.94	0.95
39. Shredded wood fiberboard, 2 in. thick on lay-in grid (mounting No. 7)	0.59	0.51	0.53	0.73	0.88	0.74	0.65
40. Thin, porous sound-absorbing material, 3/4 in. thick (mounting No. 1)	0.10	0.60	0.80	0.82	0.78	0.60	0.75
41. Thick, porous sound-absorbing material, 2 in. thick (mounting No. 1), or thin material with air space behind (mounting No. 2)	0.38	0.60	0.78	0.80	0.78	0.70	0.75
42. Sprayed cellulose fibers, 1 in. thick on concrete (mounting No. 4)	0.08	0.29	0.75	0.98	0.93	0.76	0.75

Material	Sound Absorption Coefficient						NRC
	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz	Number*
Seats and Audience(1, 3–5)‡							
43. Fabric well-upholstered seats, with perforated seat pans, unoccupied	0.19	0.37	0.56	0.67	0.61	0.59	
44. Leather-covered upholstered seats, unoccupied	0.44	0.54	0.60	0.62	0.58	0.50	
45. Audience, seated in upholstered seats§	0.39	0.57	0.80	0.94	0.92	0.87	
46. Chairs, metal or wood seats, each, unoccupied	0.15	0.19	0.22	0.39	0.38	0.30	
47. Students, informally dressed, seated in tablet-arm chairs	0.30	0.41	0.49	0.84	0.87	0.84	
Openings(5)¶							
48. Deep balcony, upholstered seats			0.50–1.00				
49. Grilles, mechanical system air			0.15–0.50				
50. Stage			0.25–0.75				
Miscellaneous(2, 5, 6)							
51. Gravel, loose and moist, 4 in. thick	0.25	0.60	0.65	0.70	0.75	0.80	0.70
52. Grass, marion bluegrass, 2 in. high	0.11	0.26	0.60	0.69	0.92	0.99	0.60
53. Snow, freshly fallen, 4 in. thick	0.45	0.75	0.90	0.95	0.95	0.95	0.90
54. Soil, rough	0.15	0.25	0.40	0.55	0.60	0.60	0.45
55. Trees, balsam firs, 20 sq ft ground area/tree, 8 ft high	0.03	0.06	0.11	0.17	0.27	0.31	0.15
56. Water surface, as in a pool	0.01	0.01	0.01	0.02	0.02	0.03	0.00

ภาคผนวก ข.
ชุดคำสั่งเบื้องหลังการพัฒนาโปรแกรม

```
Main Script Frame 1
// variables that must initial first
runray = 0;
lppointinput = 3;
lppoint = lppointinput*2;
loopmeter = 0;
metdist = 5;
loop = 10;
speedshow = 346.822;
scfac = 1;
allangle=30;
direction = 0;
precisshow = 1;
initdb = 100;
rangefromsouce=1;
timescale = 1;
// for some array that we want to keep in every stage
hittimear = [[],[],[],[],[],[],[],[],[],[],[],[],[]];
hitenergyar = [[],[],[],[],[],[],[],[],[],[],[],[],[]];
hitreflectar = [[],[],[],[],[],[],[],[],[],[],[],[],[]];
absorbco = new Array();
matname = new Array();
angle = new Array();
radian = new Array();
x = new Array();
y = new Array();
newx = new Array();
newy = new Array();
chk = new Array();
chknew = new Array();
result = new Array();
chkball = new Array();
chknewball = new Array();
resultball = new Array();
mball = new Array();
cball = new Array();
rotate = new Array();
m = new Array();
c = new Array();
alp = new Array();
energy = new Array();
initenergy = new Array();
db = new Array();
reflect = new Array();
meterx = new Array();
metery = new Array();
hitcount = new Array();
hitenergy = new Array();
hitreflect = new Array();
hittime = new Array();
```

```
hittimelast = new Array();
hitid = new Array();
hitidlast = new Array();
hitfirsttime = new Array();
timeuse = new Array();
timeusecheck= new Array();
resultdb = new Array();
hittimediffmax= new Array();
echotime= new Array();
resultrt= new Array();
s = new Array();
_root.ss = new Array();
removecheck =new Array();
// create control points, lines , material points and absorbco for each point
for (q=0; q<lppoint; q++) {
    absorbco[q] = 0;
    matname[q] = "";
    pnl1 = "p"+q;
    duplicateMovieClip("p", pnl1, q+200);
    pmn = "pm"+q;
    duplicateMovieClip("pm", pmn, q+100);
    ln = "l"+q;
    duplicateMovieClip("l", ln, q+0);
    // set control points location
    if (q<lppoint/2) {
        setProperty(pnl1, _y, 350);
        setProperty(pnl1, _x, (185+q*330/(lppoint/2-1)));
        k = q;
    }
    if (q>=lppoint/2) {
        setProperty(pnl1, _y, 530);
        setProperty(pnl1, _x, (515-(q-k-1)*330/(lppoint/2-1)));
    }
}
for (q=0; q<loopmeter; q++) {
    metem = "meter"+q;
    duplicateMovieClip("meter", metem, q+300);
    setProperty(metem, _y, 440);
    setProperty(metem, _x, 350);
}
```

Object control frame 1

```
wmode = 1;
sounddb =new Sound();
soundecho =new Sound();
soundrt =new Sound();
setProperty(ll, _visible, false);
soundcheck = 0;
gridmode = 1;
roommode = 0;
setProperty(symxplane, _visible, false);
setProperty(symyplane, _visible, false);
metmode = 3;
dbindexb = 0;
dbindex = 0;
matsampleindex = 0;
```

```
matsampleindexb = 0;
meterangle = 0;
timedelay = 50;
bestrt = 0.5;
firab=0.1;
cilab=0.1;
quality = 2;
reftime = 3;
```

....

Interface control frame 1

```
wmode =0;
genset3._visible = false;
genset5._visible = false;
genset4._visible = false;
graphicsourcebut._visible = false;
graphicrebut._visible = false;
graphicraybut._visible = false;
detailhide._visible = true;
//
anaquality.setEnabled(false);
butgraphic.gotoAndStop(2);
reftimeslide.setEnabled(false);
reftimegraphic.gotoAndStop(2);
//
status._visible = false;
direct._visible = false;
ld._visible = false;
node._visible = false;
removeg._visible = false;
//
loopslide.setValue(10);
allangslide.setValue(30);
initdbslide.setValue(100);
dbfarslide.setValue(1);
//
startbut._visible = false;
stopbut._visible = false;
cancelbut._visible = false;
//
envigraphic._visible = false;
airtempslide._visible = false;
//
//
graphicroom._visible = true;
roomlist._visible = true;
controlnum._visible = true;
symoff._visible = true;
sym1._visible = true;
sym2._visible = true;
sym3._visible = true;
//
graphicmat._visible = false;
matlist._visible = false;
matadd._visible = false;
absorbslide._visible = false;
firabslide._visible = false;
```

```
cilabslide._visible = false;
//
graphicsource._visible = false;
dbsamplelist._visible = false;
dbfarslide._visible = false;
initdbslide._visible = false;
//
graphicray._visible = false;
loopslide._visible = false;
allangslide._visible = false;
//
graphicre._visible = false;
meternumslide._visible = false;
meterdistance._visible = false;
hitareaslide._visible = false;
metmodeoff._visible = false;
meterangleslide._visible = false;
metmoderad._visible = false;
```

Main script frame 2

```
// ****************//
// for initial stage
// ****************//
if (runray == 0) {
   if (wmode == 2) {
      _root.fdbcheck();
   }
   if (wmode == 3) {
      _root.fdbcheck();
      _root.fechocheck();
   }
   _root.fresultshow();
   // clear variable
   time = 0;
   trav = 0;
   removecount = 0;
   removepercent = 0;
   setProperty(_root.removeg.removebar, _xscale, removepercent);
   initatone = Math.pow(10, -16)*(Math.pow(10, (initdb/10)));
   initall = 4*Math.PI*Math.pow(rangefromsouce*100, 2)*initatone;
   // create all balls and set properties
   startang = direction-allangle/2;
   angdif = allangle/loop;
   allangle = angdif*loop;
   nodex = Number(getProperty(node, _x));
   nodey = Number(getProperty(node, _y));
   for (i=0; i<loop; i++) {
      bn = "ball"+i;
      duplicateMovieClip("ball", bn, i+4000);
      setProperty(bn, _x, nodex);
      setProperty(bn, _y, nodey);
      angle[i] = -startang-angdif*i;
      initenergy[i] = initall;
```

```
        alp[i] = 100;
        reflect[i] = 0;
        removecheck[i] = 0;
    }
}
//
// ****************//
// for proceesing stage
// ****************//
if (runray == 1) {
    // set next time and travel lenght
    time = time+1;
    trav = time*speedshow/1000;
    if (removecount == loop) {
        runray = 0;
    }
    removecount = 0;
    //
    // make balls move//
    speed = speedshow*scfac/100;
    for (i=0; i<loop; i++) {
        removecount = removecount+removecheck[i];
        bn = "ball" add i;
        radiani = angle[i]*0.0174532925;
        xi = Number(getProperty(bn, _x));
        yi = Number(getProperty(bn, _y));
        newxi = xi+speed*Math.cos(radiani);
        newyi = yi+speed*Math.sin(radiani);
        mballi = (newyi-yi)/(newxi-xi);
        cballi = ((yi*newxi)-(newyi*xi))/(newxi-xi);
        setProperty(bn, _x, newxi);
        setProperty(bn, _y, newyi);
        //
        if (trav == 0 || trav == "") {
            // energy[i] = initenergy[i];
        }
        if (trav>0) {
            energyi = initenergy[i]/(4*Math.PI*Math.pow(trav*100, 2));
            energymin = Math.pow(10, -16);
            en = energyi/energymin;
            db[i] = 10*(Math.log(Math.E)/Math.log(10))*(Math.log(en));
            alpi = db[i];
            // set energy of balls and show in alpha
            setProperty(bn, _alpha, alpi);
            // clear balls if energy less than "0"
            if (db[i]<0) {
                db[i] = 0;
                removeMovieClip(bn);
                removecheck[i] = 1;
            }
        }
        if (removecount == loop) {
            runray = 0;
        }
        if (wmode == 3) {
```

```
        if (reflect[i]>=reftime) {
            reflect[i] = 0;
            removeMovieClip(bn);
            removecount = removecount+1;
        }
    }
    //
    // check balls hit main boundary then remove
    if (newyi>=620 || newyi<=260 || newxi>=680 || newxi<=20) {
        removeMovieClip(bn);
        removecheck[i] = 1;
    }
    //
    // check that balls hit walls then change direction//
    for (j=0; j<lppoint; j++) {
        pn1 = "p"+j;
        k = j+1;
        pn2 = "p"+k;
        pnx1 = Number(getProperty(pn1, _x));
        pny1 = Number(getProperty(pn1, _y));
        pnx2 = Number(getProperty(pn2, _x));
        pny2 = Number(getProperty(pn2, _y));
        if (j == lppoint-1) {
            pnx2 = Number(getProperty(p0, _x));
            pny2 = Number(getProperty(p0, _y));
        }
        // find rotation of wall in degree
        if (pnx2>=pnx1 && pny2>=pny1) {
            rotatej = Math.atan(Math.abs((pny2-pny1)/(pnx2-pnx1)))*57.29577951;
        }
        if (pnx2<=pnx1 && pny2>=pny1) {
            rotatej = 180-Math.atan(Math.abs((pny2-pny1)/(pnx2-pnx1)))*57.29577951;
        }
        if (pnx2<=pnx1 && pny2<=pny1) {
            rotatej = 180+Math.atan(Math.abs((pny2-pny1)/(pnx2-pnx1)))*57.29577951;
        }
        if (pnx2>=pnx1 && pny2<=pny1) {
            rotatej = 360-Math.atan(Math.abs((pny2-pny1)/(pnx2-pnx1)))*57.29577951;
        }
        // check line crossing
        mj = (pny2-pny1)/(pnx2-pnx1);
        cj = ((pny1*pnx2)-(pny2*pnx1))/(pnx2-pnx1);
        if (isFinite(mj) == true && isFinite(mballj) == true) {
            chki = (mj*xi)+cj-yi;
            chknewi = (mj*newxi)+cj-newyi;
            resulti = chknewi/chki;
            chkballi = (mballi*pnx1)+cballi-pny1;
            chknewballi = (mballi*pnx2)+cballi-pny2;
            resultballi = chknewballi/chkballi;
            // ( if m=0 make result be infite so it musthave special condition
        } else if (isFinite(mj) == false) {
            chki = pnx1-xi;
            chknewi = pnx1-newxi;
            resulti = chknewi/chki;
            chkballi = (mballi*pnx1)+cballi-pny1;
            chknewballi = (mballi*pnx2)+cballi-pny2;
            resultballi = chknewballi/chkballi;
        } else if (isFinite(mballi) == false) {
            chki = (mj*xi)+cj-yi;
            chknewi = (mj*newxi)+cj-newyi;
            resulti = chknewi/chki;
            chkballi = pnx1-xi;
            chknewballi = pnx2-xi;
            resultballi = chknewballi/chkballi;
        }
        // if cross then change ball direction
        if (resulti<=0 && resultballi<=0) {
            setProperty(bn, _x, xi);
            setProperty(bn, _y, yi);
            angle[i] = (360+(2*rotatej))-angle[i];
            initenergy[i] = initenergy[i]-(initenergy[i]*absorbco[j]);
            reflect[i] = reflect[i]+1;
            tellTarget (bn) {
                nextFrame();
            }
        }
    }
}
//
// check ball hit meter point
for (i=0; i<loop; i++) {
    bn = "ball"+i;
    xi = Number(getProperty(bn, _x));
    yi = Number(getProperty(bn, _y));
    timeusecheckall = 0;
    for (im=0; im<loopmeter; im++) {
        meterm = "meter"+im;
        meterxim = Number(getProperty(meterm, _x));
        meteryim = Number(getProperty(meterm, _y));
        if (wmode == 2) {
            if (hitfirsttime[im]>0) {
                timeuse[im] = time-hitfirsttime[im];
                if (timeuse[im]>80) {
                    timeusecheck[im] = 1;
                }
            }
            timeusecheckall = timeusecheckall+timeusecheck[im];
        }
        if (Math.abs(meterxim-xi)<=precis/2 && Math.abs(meteryim-yi)<=precis/2) {
            // ball mush not the same one on last hit
            hitid[im] = i;
            if (hitid[im] !== hitidlast[im] && hittimelast[im] != time && hittimelast[im]<=time-3) {
                hitcount[im] = hitcount[im]+1;
                hitenergy[im] = db[i];
                hittime[im] = time;
                hitreflect[im] = reflect[i];
                if (hitcount[im] == 1) {
                    hitfirsttime[im] = time;
                }
                hitenergyar[im][s[im]] = hitenergy[im];
```

```
                    hittimear[im][s[im]] = hittime[im];
                    hitreflectar[im][s[im]] = hitreflect[im];
                    s[im] = s[im]+1;
                    _root.ss[im] = s[im];
                    hitidlast[im] = hitid[im];
                    hittimelast[im] = time;
                    _root.fsettimeline();
                    _root.fhitshow();
                }
            }
        }
        if (wmode == 2) {
            if (timeusecheckall == loopmeter) {
                runray = 0;
            }
        }
    }
    if (wmode == 3) {
        removepercent = removecount*100/loop;
        setProperty(_root.removeg.removebar, _xscale, removepercent);
    }
}
function fdbcheck() {
    for (i=0; i<loopmeter; i++) {
        hitdbold = 0;
        for (ii=0; ii<_root.ss[i]; ii++) {
            hittimedb = hittimear[i][ii]-hittimear[i][0];
            hitdbnew = hitenergyar[i][ii];
            if (hittimedb<timedelay) {
                if (hitdbold == 0) {
                    resultdb[i] = hitdbnew;
                }
                if (hitdbold>0) {
                    wattold = Math.pow(10, -16)*(Math.pow(10, (hitdbold/10)));
                    wattnew = Math.pow(10, -16)*(Math.pow(10, (hitdbnew/10)));
                    wattmix = wattold+wattnew;
                    wattmin = Math.pow(10, -16);
                    wattresult = wattmix/wattmin;
                    resultdb[i] = 10*(Math.log(Math.E)/Math.log(10))*(Math.log(wattresult));
                }
            }
            hitdbold = resultdb[i];
        }
    }
}
function fechocheck() {
    for (i=0; i<loopmeter; i++) {
        echotime[i] = 0;
        hittimediffmax[i] = 0;
        for (ii=0; ii<_root.ss[i]; ii++) {
            if (ii>0&&hitreflectar[i][ii]<3) {
                hittimediff = hittimear[i][ii]-hittimear[i][ii-1];
                if (hittimediff>=60) {
                    echotime[i] = echotime[i]+1;
                }
                if (hittimediff>hittimediffmax[i]) {
                    hittimediffmax[i] = hittimediff;
                }
            }
        }
    }
}
function frtcheck() {
    for (i=0; i<loopmeter; i++) {
        resultrt[i] = 0;
        diffenergy1 = hitenergyar[i][0]-5;
        diffenergy2 = hitenergyar[i][0]-35;
        for (ii=0; ii<_root.ss[i]; ii++) {
            checkenergy = hitenergyar[i][ii];
            if (checkenergy>=diffenergy1) {
                firsttime = hittimear[i][ii];
            }
            if (checkenergy>=diffenergy2) {
                lasttime = hittimear[i][ii];
            }
            resultrt[i] = (lasttime-firsttime)*2;
            if (resultrt[i] == 0) {
                resultrt[i] = "NO Answer";
            }
        }
    }
}
function fhitshow() {
    hitidshow = hitid[meterindex];
    hitidlastshow = hitidlast[meterindex];
    hitcountshow = hitcount[meterindex];
    hitenergyshow = hitenergy[meterindex];
    hittimeshow = hittime[meterindex];
    hitreflectshow = hitreflect[meterindex];
    hitfirsttimeshow = hitfirsttime[meterindex];
    timeuseshow = timeuse[meterindex];
    //
}
function fresultshow() {
    resultdbshow = resultdb[meterindex];
    if (resultdbshow!="") {
        firsthitdbshow = hitenergyar[meterindex][0];
        diffdb = resultdbshow-firsthitdbshow;
    }
    //
    hittimediffmaxshow = hittimediffmax[meterindex];
    echotimeshow = echotime[meterindex];
    rtshow = resultrt[meterindex];
    //
    if (diffdb==0||diffdb==""||isFinite(diffdb)==false) {
        icondb.gotoAndStop(1);
    }
    if (diffdb>3) {
        icondb.gotoAndStop(2);
    }
```

```
  if (diffdb<=3&&diffdb>0) {
    icondb.gotoAndStop(3);
  }
  if (hittimediffmaxshow==""||hittimediffmaxshow==0||isFinite(hittimediffmaxshow)==false) {
    iconecho.gotoAndStop(1);
  }
  if (hittimediffmaxshow>0&&hittimediffmaxshow<=35) {
    iconecho.gotoAndStop(2);
  }
  if (hittimediffmaxshow>35&&hittimediffmaxshow<=60) {
    iconecho.gotoAndStop(3);
  }
  if (hittimediffmaxshow>60) {
    iconecho.gotoAndStop(4);
  }
  if (diffrt>=0.0&&diffrt<0.2) {
    iconrt.gotoAndStop(2);
  }
  if (diffrt>=0.2&&diffrt<0.4) {
    iconrt.gotoAndStop(3);
  }
  if (diffrt>=0.4) {
    iconrt.gotoAndStop(4);
  }
  if (isFinite(diffrt)==false) {
    iconrt.gotoAndStop(1);
  }
}
```

Object Control frame 2

```
function fchkcontroldrag() {
  // set control points if symetry mode is on
  if (roommode == 0) {
  }
  if (roommode == 1) {
    for (i=0; i<lppoint; i++) {
      pn = "p"+i;
      pnn = "p"+(lppoint-(i+1));
      setProperty(pnn, _y, (440-Number(getProperty(pn, _y)))+440);
      setProperty(pnn, _x, Number(getProperty(pn, _x)));
    }
  }
  if (roommode == 2) {
    for (i=0; i<lppoint/2; i++) {
      pn = "p"+i;
      pnn = "p"+(lppoint/2-i-1);
      if (pn != pnn) {
        setProperty(pnn, _x, (350-Number(getProperty(pn, _x)))+350);
        setProperty(pnn, _y, Number(getProperty(pn, _y)));
      }
      if (pn == pnn) {
        setProperty(pnn, _x, 350);
      }
      pm = "p"+(lppoint-i-1);
      pmm = "p"+(lppoint/2+i);
```

```
            if (pm != pmm) {
               setProperty(pmm, _x, (350-Number(getProperty(pm, _x)))+350);
               setProperty(pmm, _y, Number(getProperty(pm, _y)));
            }
            if (pm == pmm) {
               setProperty(pmm, _x, 350);
            }
         }
      }
      if (roommode == 3) {
         for (i=0; i<lppoint; i++) {
            pn = "p"+i;
            pm = "p"+(lppoint/2-(i+1));
            if (pn != pm) {
               setProperty(pm, _x, (350-Number(getProperty(pn, _x)))+350);
               setProperty(pm, _y, Number(getProperty(pn, _y)));
            }
            if (pn == pm) {
               setProperty(pm, _x, 350);
            }
            pnn = "p"+(lppoint-(i+1));
            setProperty(pnn, _y, (440-Number(getProperty(pn, _y)))+440);
            setProperty(pnn, _x, Number(getProperty(pn, _x)));
         }
      }
   }
   //
   function fsetlinemat() {
      // set position & properties of lines , material points and absorbco
      for (q=0; q<lppoint; q++) {
         pnl1 = "p"+q;
         r = q+1;
         pnl2 = "p"+r;
         pnlx1 = Number(getProperty(pnl1, _x));
         pnly1 = Number(getProperty(pnl1, _y));
         pnlx2 = Number(getProperty(pnl2, _x));
         pnly2 = Number(getProperty(pnl2, _y));
         if (q == lppoint-1) {
            pnlx2 = Number(getProperty(p0, _x));
            pnly2 = Number(getProperty(p0, _y));
         }
         ln = "l"+q;
         setProperty(ln, _x, pnlx1);
         setProperty(ln, _y, pnly1);
         setProperty(ln, _xscale, pnlx2-pnlx1);
         setProperty(ln, _yscale, pnly2-pnly1);
         setProperty(ln, _alpha, 100-(absorbco[q]*100));
         pmn = "pm"+q;
         setProperty(pmn, _x, (pnlx2+pnlx1)/2);
         setProperty(pmn, _y, (pnly2+pnly1)/2);
      }
   }
   function fchkmeterdrag() {
      // set meter if meter mode is on
      if (metmode == 3) {
        for (i=0; i<loopmeter; i++) {
            metem = "meter"+i;
            meterradian = meterangle*0.0174532925;
            setProperty(metem, _x, (getProperty(node, _x))+(Math.cos(meterradian)*metdist*10*scfac*(i+1)));
            setProperty(metem, _y, (getProperty(node, _y))+(Math.sin(meterradian)*metdist*10*scfac*(i+1)));
        }
    }
    precis = precisshow*10;
    for (w=0; w<loopmeter; w++) {
        metem = "meter"+w;
        setProperty(metem, _width, precis);
        setProperty(metem, _height, precis);
    }
}
//
function fsetmeter() {
    // set meter parameter
    metemn = "meter"+[meterindex];
    metx = Number(getProperty(metemn, _x));
    mety = Number(getProperty(metemn, _y));
    mmetx = (metx-350)/(10*scfac);
    mmety = -(mety-440)/(10*scfac);
    nodex = Number(getProperty(node, _x));
    nodey = Number(getProperty(node, _y));
    meterx = Number(getProperty(metemn, _x));
    metery = Number(getProperty(metemn, _y));
    if (meterindex == "" || wmode == 0 || mode == 1) {
        setProperty(ll, _visible, false);
    }
    if (meterindex != "" && wmode != 0 && wmode != 1) {
        setProperty(ll, _visible, true);
        setProperty(ll, _x, node._x);
        setProperty(ll, _y, node._y);
        setProperty(ll, _xscale, meterx-node._x);
        setProperty(ll, _yscale, metery-node._y);
    }
    meterfar = Math.round(Math.sqrt(Math.pow((meterx-nodex), 2)+Math.pow((metery-nodey), 2))/(scfac*10)*10)/10;
    if (nodex>=meterx && nodey>=metery) {
        meterang = Math.round(180-Math.atan(Math.abs((metery-nodey)/(meterx-nodex)))*57.29577951);
    }
    if (nodex<=meterx && nodey>=metery) {
        meterang = Math.round(Math.atan(Math.abs((metery-nodey)/(meterx-nodex)))*57.29577951);
    }
    if (nodex<=meterx && nodey<=metery) {
        meterang = Math.round(-Math.atan(Math.abs((metery-nodey)/(meterx-nodex)))*57.29577951);
    }
    if (nodex>=meterx && nodey<=metery) {
        meterang = Math.round(-180+Math.atan(Math.abs((metery-nodey)/(meterx-nodex)))*57.29577951);
    }
}
//
function fsetdirect() {
    // set direction parameter
    directx = Number(getProperty(direct, _x));
    directy = Number(getProperty(direct, _y));
```

```
   setProperty(ld, _x, node._x);
   setProperty(ld, _y, node._y);
   setProperty(ld, _xscale, direct._x-node._x);
   setProperty(ld, _yscale, direct._y-node._y);
   if (nodex>=directx && nodey>=directy) {
      direction = Math.round(180-Math.atan(Math.abs((directy-nodey)/(directx-nodex)))*57.29577951);
   }
   if (nodex<=directx && nodey>=directy) {
      direction = Math.round(Math.atan(Math.abs((directy-nodey)/(directx-nodex)))*57.29577951);
   }
   if (nodex<=directx && nodey<=directy) {
      direction = Math.round(-Math.atan(Math.abs((directy-nodey)/(directx-nodex)))*57.29577951);
   }
   if (nodex>=directx && nodey<=directy) {
      direction = Math.round(-180+Math.atan(Math.abs((directy-nodey)/(directx-nodex)))*57.29577951);
   }
   //
}
function fchecksource() {
   sx = Number(getProperty(node, _x));
   sy = Number(getProperty(node, _y));
   sourcex = (sx-350)/(10*scfac);
   sourcey = -(sy-440)/(10*scfac);
}
//
function fcheckarea() {
   xall = 0;
   yall = 0;
   for (j=0; j<lppoint; j++) {
      pq = "p"+j;
      xq = Number(getProperty(pq, _x));
      yq = Number(getProperty(pq, _y));
      xall = xall+xq;
      yall = yall+yq;
   }
   xtarget = xall/lppoint;
   ytarget = yall/lppoint;
   // setProperty(_root.t, _x, xtarget);
   // setProperty(_root.t, _y, ytarget);
   //
   areaold = 0;
   totalold = 0;
   for (i=0; i<lppoint; i++) {
      pn = "p"+i;
      pnn = "p"+(i+1);
      // zn = "z"+i;
      if (i == lppoint-1) {
         pnn = "p0";
      }
      xn = Number(getProperty(pn, _x));
      yn = Number(getProperty(pn, _y));
      xnn = Number(getProperty(pnn, _x));
      ynn = Number(getProperty(pnn, _y));
      xt = Number(getProperty(t, _x));
      yt = Number(getProperty(t, _y));
```

```
        m = (ynn-yn)/(xnn-xn);
        c = ((yn*xnn)-(ynn*xn))/(xnn-xn);
        mt = (-1)/m;
        ct = yt-(mt*xt);
        xo = (c-ct)/(mt-m);
        yo = ((m*ct)-(mt*c))/(m-mt);
        if (isFinite(m) == false || mt == 0) {
            xo = xn;
            yo = yt;
        }
        if (m == 0 || isFinite(mt) == false) {
            xo = xt;
            yo = yn;
        }
        // setProperty(_root.zn, _x, xo);
        // setProperty(_root.zn, _y, yo);
        lenght = (Math.sqrt(Math.pow((xnn-xn), 2)+Math.pow((ynn-yn), 2)))/10;
        hight = (Math.sqrt(Math.pow((xt-xo), 2)+Math.pow((yt-yo), 2)))/10;
        area = (lenght/scfac)*(hight/(2*scfac));
        areaall = areaold+area;
        volumnall = areaall*depthall;
        areaold = areaall;
        //
        panelarea = (lenght/scfac)*depthall;
        panelabsorp = absorbco[i]*panelarea;
        totalabsorp = totalold+panelabsorp;
        totalold = totalabsorp;
        //
        totalabsorpz = totalabsorp+(areaall*flrab)+(areaall*cilab);
        rtall = 0.16*volumnall/(totalabsorpz);
        diffrt = Math.abs(rtall-bestrt);
    }
}
if (runray == 0) {
    _root.fchkcontroldrag();
    _root.fchkmeterdrag();
    _root.fsetlinemat();
    _root.fsetmeter();
    _root.fsetdirect();
    _root.fchecksource();
    _root.fcheckarea();
}
//
function dragpoint(pid) {
    if (roomsampleindex==0) {
        roomlist.setSelectedIndex(0);
    }
    genset1.setState(true);
    _root.fgenset();
    if (runray==0) {
        startDrag(pid, false, 20, 260, 680, 620);
    }
}
function pressmatpoint(mid) {
    abindex = mid;
```

```
   lineindex = mid;
   absorbshow = absorbco[mid];
   absorbslide.setValue(absorbshow);
   matnameshow = matname[mid];
   genset2.setState(true);
   _root.fgenset();
   // set select line display in red
   for (i=0; i<lppoint; i++) {
      ln = "l"+i;
      lx = "l"+mid;
      tellTarget (ln) {
         gotoAndStop(1);
      }
      tellTarget (lx) {
         gotoAndStop(2);
      }
   }
}
function clearmatpoint() {
   absorbshow = "";
   matnameshow = "";
   abindex = "";
   lineindex = "";
   for (i=0; i<lppoint; i++) {
      ln = "l"+i;
      tellTarget (ln) {
         gotoAndStop(1);
      }
   }
}
function dragnode() {
   if (wmode==1) {
      genset4.setState(true);
   }
   if (wmode==2 || wmode==3 || wmode==4|| wmode==5) {
      genset3.setState(true);
   }
   _root.fgenset();
   if (runray==0) {
      startDrag(node, false, 20, 260, 680, 620);
   }
}
function dragdirect() {
   genset4.setState(true);
   _root.fgenset();
   if (runray==0) {
      startDrag(direct, false, 20, 260, 680, 620);
   }
}
function dragmeter(metid) {
   genset5.setState(true);
   _root.fgenset();
   metern = "meter"+metid;
   if (runray==0) {
      startDrag(metern, false, 20, 260, 680, 620);
```

```
  }
  // set select meter display in red
  for (i=0; i<loopmeter; i++) {
    mn = "meter"+i;
    mx = "meter"+metid;
    tellTarget (mn) {
      gotoAndStop(1);
    }
    tellTarget (mx) {
      gotoAndStop(2);
    }
    meterindexold = meterindex;
    meterindex = metid;
    _root.fsettimeline();
    _root.fhitshow();
    _root.meterdetailhide._visible = false;
  }
  if (runray==1) {
    _root.fsetmeter();
  }
}
function clearmeter() {
  for (i=0; i<loopmeter; i++) {
    mn = "meter"+i;
    tellTarget (mn) {
      gotoAndStop(1);
    }
  }
  meterindexold = meterindex;
  meterindex = "";
  _root.fsettimeline();
  _root.fhitshow();
  _root.meterdetailhide._visible = true;
}
// function for press and release on each control point
node.onPress = function() {
  dragnode();
};
node.onRelease = function() {
  stopDrag();
};
node.onReleaseOutside = function() {
  stopDrag();
};
direct.onPress = function() {
  dragdirect(0);
};
direct.onRelease = function() {
  stopDrag();
};
direct.onReleaseOutside = function() {
  stopDrag();
}.
// function for press and release on each control point
meter0.onPress = function() {
   dragmeter(0);
};
meter0.onRelease = function() {
   stopDrag();
};
meter0.onReleaseOutside = function() {
   stopDrag();
};
meter1.onPress = function() {
   dragmeter(1);
};
meter1.onRelease = function() {
   stopDrag();
};
meter1.onReleaseOutside = function() {
   stopDrag();
};
....
meter11.onPress = function() {
   dragmeter(11);
};
meter11.onRelease = function() {
   stopDrag();
};
meter11.onReleaseOutside = function() {
   stopDrag();
};
...
p0.onPress = function() {
   dragpoint(p0);
};
p0.onRelease = function() {
   stopDrag();
};
p0.onReleaseOutside = function() {
   stopDrag();
}
...;
p20.onPress = function() {
   dragpoint(p20);
};
p20.onRelease = function() {
   stopDrag();
};
p20.onReleaseOutside = function() {
   stopDrag();
};
...
//
// function for press on each material point
pm0.onPress = function() {
   pressmatpoint(0);
};
pm1.onPress = function() {
   pressmatpoint(1);
```

```
};
...
pm20.onPress = function() {
   pressmatpoint(20);
};
```

Interface Control frame 2

```
function fstart() {
   runray = 1;
}
function fstop() {
   runray = 0;
}
function freset() {
   for (i=0; i<loopmeter; i++) {
      for (j=0; j<_root.ss[i]; j++) {
         hitenergyar[i][j] = "";
         hittimear[i][j] = "";
         hitreflectar[i][j] = "";
      }
   }
   for (im=0; im<loopmeter; im++) {
      hitcount[im] = 0;
      hitid[im] = "";
      hitidlast[im] = "";
      hittime[im] = "";
      hittimelast[im] = "";
      hitenergy[im] = "";
      timeuse[im] = "";
      hitfirsttime[im] = "";
      s[im] = 0;
      resultdb[im] = "";
      //
      hittimediffmax[im] = "";
      echotime[im] = "";
      resultrt[im] = "";
   }
   _root.fsettimeline();
   _root.fhitshow();
   _root.fresultshow();
}
function setsymplane() {
   if (roommode == 0) {
      setProperty(symxplane, _visible, false);
      setProperty(symyplane, _visible, false);
   }
   if (roommode == 1) {
      setProperty(symxplane, _visible, true);
      setProperty(symyplane, _visible, false);
   }
   if (roommode == 2) {
      setProperty(symxplane, _visible, false);
      setProperty(symyplane, _visible, true);
   }
   if (roommode == 3) {
      setProperty(symxplane, _visible, true);
      setProperty(symyplane, _visible, true);
   }
}
function fsymetry() {
   if (roomsampleindex == 0) {
      roomlist.setSelectedIndex(0);
   }
   if (symoff.getState() == true) {
      roommode = 0;
   }
   if (sym1.getState() == true) {
      roommode = 1;
   }
   if (sym2.getState() == true) {
      roommode = 2;
   }
   if (sym3.getState() == true) {
      roommode = 3;
   }
   _root.setsymplane();
}
function fcontrolnum(component) {
   if (roomsampleindex == 0) {
      roomlist.setSelectedIndex(0);
   }
   lppointinput = controlnum.getValue();
   // remove all control points , lines , material points
   for (w=0; w<lppoint; w++) {
      pnl1 = "p"+w;
      ln = "l"+w;
      pmn = "pm"+w;
      removeMovieClip(pnl1);
      removeMovieClip(ln);
      removeMovieClip(pmn);
   }
   // create control points, lines , material points and absorbco for each point
   lppoint = lppointinput*2;
   for (q=0; q<lppoint; q++) {
      absorbco[q] = 0;
      matname[q] = "";
      pnl1 = "p"+q;
      duplicateMovieClip("p", pnl1, q+200);
      pmn = "pm"+q;
      duplicateMovieClip("pm", pmn, q+100);
      ln = "l"+q;
      duplicateMovieClip("l", ln, q+0);
      // set control points location
      if (q<lppoint/2) {
         setProperty(pnl1, _y, 350);
         setProperty(pnl1, _x, (185+q*330/(lppoint/2-1)));
         k = q;
      }
      if (q>=lppoint/2) {
            setProperty(pnl1, _y, 530);
            setProperty(pnl1, _x, (515-(q-k-1)*330/(lppoint/2-1)));
        }
    }
}
function fscfac(component) {
    _root.fmeternum();
    if (wmode == 1||wmode == 0) {
        _root.fmeterhide();
    }
    scfac = scfacslide.getValue();
    // set scale bar & grid display to scale factor
    if (gridmode == 0) {
        setProperty(_root.scalebar, _width, 100*scfac);
    }
    if (gridmode == 1) {
        setProperty(_root.scalebar, _width, 100*scfac);
        if (scfac>0.8) {
            setProperty(_root.grid, _xscale, scfac*100);
            setProperty(_root.grid, _yscale, scfac*100);
        }
        if (scfac<=0.8) {
            setProperty(_root.grid, _xscale. scfac*1000);
            setProperty(_root.grid, _yscale, scfac*1000);
        }
    }
}
function fgridcheck(component) {
    if (gridon.getState() == true) {
        gridmode = 1;
        setProperty(_root.grid, _visible, true);
        setProperty(_root.gridgraphic, _visible, true);
    }
    if (gridoff.getState() == true) {
        gridmode = 0;
        setProperty(_root.grid, _visible, false);
        setProperty(_root.gridgraphic, _visible, false);
    }
}
function floopslide(component) {
    loop = loopslide.getValue();
}
function fallangslide(component) {
    allangle = allangslide.getValue();
}
function fabsorbslide(component) {
    if (matsampleindex == 0) {
        matsampleindexb = 1;
        absorbshow = absorbslide.getValue();
        absorbco[abindex] = absorbshow;
        matlist.setSelectedIndex(0);
        matsampleindexb = 0;
        matadd.setEnabled(true);
    }
}
```

```
graphicmat.matnametext.onChanged = function() {
    matname[abindex] = matnameshow;
    matadd.setEnabled(true);
};
function fmatlist(component) {
    if (matsampleindexb == 0) {
        matsampleindex = 1;
        matnameshow = matList.getSelectedItem().label;
        absorbshow = matList.getSelectedItem().data;
        absorbslide.setValue(absorbshow);
        matname[abindex] = matnameshow;
        absorbco[abindex] = absorbshow;
        matsampleindex = 0;
    }
}
function fmatadd(component) {
    matlist.addItem(matnameshow, absorbshow);
    ml = matlist.getLength()-1;
    matlist.setSelectedIndex(ml);
}
function fmeterhide(component) {
    _root.clearmeter();
    for (q=0; q<loopmeter; q++) {
        metem = "meter"+q;
        setProperty(metem, _visible, false);
    }
}
function fmetershow(component) {
    _root.clearmeter();
    for (q=0; q<loopmeter; q++) {
        metem = "meter"+q;
        setProperty(metem, _visible, false);
    }
}
function fmeterremove(component) {
    for (q=0; q<loopmeter; q++) {
        metem = "meter"+q;
        removeMovieClip(metem);
    }
}
function fmeternum(component) {
    _root.fmeterremove();
    loopmeter = meternumslide.getValue();
    for (q=0; q<loopmeter; q++) {
        metem = "meter"+q;
        duplicateMovieClip("meter", metem, q+300);
    }
    for (i=0; i<loopmeter; i++) {
        metem = "meter"+i;
        if (metmode == 0) {
            setProperty(metem, _y, (getProperty(node, _y)));
            setProperty(metem, _x, (getProperty(node, _x))+(10*scfac*precisshow*(i+1)));
        }
        if (metmode == 3) {
            meterradian = meterangle*0.0174532925;
```

```
         setProperty(metem, _x, (getProperty(node, _x))+(Math.cos(meterradian)*metdist*10*scfac*(i+1)));
         setProperty(metem, _y, (getProperty(node, _y))+(Math.sin(meterradian)*metdist*10*scfac*(i+1)));
      }
   }
   for (i=0; i<loopmeter; i++) {
      mn = "meter"+i;
      mx = "meter"+meterindex;
      tellTarget (mn) {
         gotoAndStop(1);
      }
      tellTarget (mx) {
         gotoAndStop(2);
      }
   }
}
function fmetmode(component) {
   if (metmodeoff.getState() == true) {
      metmode = 0;
   }
   if (metmoderad.getState() == true) {
      metmode = 3;
   }
}
function myHandler15(component) {
   metdist = meterdistance.getValue();
}
function myHandler6(component) {
   precisshow = hitareaslide.getValue();
}
function fmeterangle() {
   meterangle = -meterangleslide.getValue();
}
timeplus.setClickHandler("ftimeplus");
function ftimeplus(component) {
   timescale = timescale*2;
   timenum1 = 100/timescale;
   timenum2 = 200/timescale;
   _root.fsettimeline();
}
timeminus.setClickHandler("ftimeminus");
function ftimeminus(component) {
   timescale = timescale/2;
   timenum1 = 100/timescale;
   timenum2 = 200/timescale;
   _root.fsettimeline();
}
copytime.setClickHandler("fcopytime");
function fcopytime(component) {
   for (w=0; w<_root.ss[meterindex]; w++) {
      barn = "bar"+w;
      removeMovieClip(barn);
   }
   for (w=0; w<_root.ssgray; w++) {
      bargrayn = "bargray"+w;
      removeMovieClip(bargrayn);
   }
   for (w=0; w<_root.ss[meterindex]; w++) {
      bargrayn = "bargray"+w;
      duplicateMovieClip("_root.bargray", bargrayn, w+2000);
      setProperty(bargrayn, _height, hitenergyar[meterindex](w]);
      setProperty(bargrayn, _x, (hittimear[meterindex][w]*timescaie)+740);
      setProperty(bargrayn, _y, 662);
      _root.ssgray = _root.ss[meterindex];
      meterindexcopy = meterindex;
   }
   _root.fsettimeline();
}
cleartime.setClickHandler("fcleartime");
function fcleartime(component) {
   for (w=0; w<_root.ss[meterindex]; w++) {
      barn = "bar"+w;
      removeMovieClip(barn);
   }
   meterindexold = "";
   meterindex = "";
   _root.fsettimeline();
}
clearcopy.setClickHandler("fclearcopy");
function fclearcopy(component) {
   for (w=0; w<_root.ssgray; w++) {
      bargrayn = "bargray"+w;
      removeMovieClip(bargrayn);
   }
   _root.fsettimeline();
}
function fsettimeline() {
   for (w=0; w<_root.ss[meterindexold]; w++) {
      barn = "bar"+w;
      removeMovieClip(barn);
   }
   for (w=0; w<_root.ss[meterindex]; w++) {
      barn = "bar"+w;
      duplicateMovieClip("_root.bar", barn, w+3000);
      setProperty(barn, _height, hitenergyar[meterindex][w]);
      setProperty(barn, _x, (hittimear[meterindex][w]*timescale)+740);
      setProperty(barn, _y, 662);
      framenum = hitreflectar[meterindex][w];
      if (framenum==0) {
         tellTarget (barn) {
            gotoAndStop(1);
         }
      }
      if (framenum==1) {
         tellTarget (barn) {
            gotoAndStop(2);
         }
      }
      if (framenum==2) {
         tellTarget (barn) {
            gotoAndStop(3);
            }
        }
        if (framenum==3) {
            tellTarget (bam) {
                gotoAndStop(4);
            }
        }
        if (framenum==4) {
            tellTarget (bam) {
                gotoAndStop(5);
            }
        }
        if (framenum==5) {
            tellTarget (bam) {
                gotoAndStop(6);
            }
        }
        if (framenum==6) {
            tellTarget (bam) {
                gotoAndStop(7);
            }
        }
        if (framenum==7) {
            tellTarget (bam) {
                gotoAndStop(8);
            }
        }
        if (framenum>7) {
            tellTarget (bam) {
                gotoAndStop(9);
            }
        }
    }
    for (w=0; w<_root.ssgray; w++) {
        bargrayn = "bargray"+w;
        setProperty(bargrayn, _x, (hittimear[meterindexcopy][w]*timescale)+740);
    }
}
function fairtemp(component) {
    airtemp = airtempslide.getValue();
    speedshow = 331.4*Math.sqrt(1+airtemp/273);
}
function fdbsample() {
    if (dbindexb == 0) {
        dbindex = 1;
        initdb = dbsamplelist.getSelectedItem().data;
        rangefromsouce = 1;
        initdbslide.setValue(initdb);
        dbfarslide.setValue(rangefromsouce);
        dbindex = 0;
    }
}
function fdbfar(component) {
    if (dbindex == 0) {
        dbindexb = 1;
        rangefromsouce = dbfarslide.getValue();
        dbsamplelist.setSelectedIndex(0);
        dbindexb = 0;
    }
}
function finitdb(component) {
    if (dbindex == 0) {
        dbindexb = 1;
        initdb = initdbslide.getValue();
        dbsamplelist.setSelectedIndex(0);
        dbindexb = 0;
    }
}
function fanaquality() {
    quality = anaquality.getValue();
    loopslide.setValue(45*quality);
}
function fsourcetype() {
    if (butspeech.getState() == true) {
        timedelay = 50;
    }
    if (butmusic.getState() == true) {
        timedelay = 80;
    }
    _root.fdbcheck();
    _root.fresultshow();
}
function froomuse(component) {
    bestrt = roomuselist.getSelectedItem().data;
}
function fflrab(component) {
    flrab = flrabslide.getValue();
}
function fcilab(component) {
    cilab = cilabslide.getValue();
}
function freftime(component) {
    reftime = reftimeslide.getValue();
}

function fwmode() {
    if (workmode0.getState() == true) {
        wmode =0;
        _root.freset();
        //
        genset3._visible = false;
        genset5._visible = false;
        genset4._visible = false;
        graphicsourcebut._visible = false;
        graphicrebut._visible = false;
        graphicraybut._visible = false;
        detailhide._visible = true;
        //
        anaquality.setEnabled(false);
        butgraphic.gotoAndStop(2);
    reftimeslide.setEnabled(false);
    reftimegraphic.gotoAndStop(2);
    //
    status._visible = false;
    removeg._visible = false;
    direct._visible = false;
    ld._visible = false;
    node._visible = false;
    //
    _root.clearmeter();
    _root.fmeterhide();
    //
    loopslide.setValue(10);
    allangslide.setValue(30);
    initdbslide.setValue(100);
    dbfarslide.setValue(1);
    //
    startbut._visible = false;
    stopbut._visible = false;
    cancelbut._visible = false;
    //
    envigraphic._visible = false;
    airtempslide._visible = false;
    //
    genset1.setState(true);
    _root.fgenset();
    //
}
if (workmode1.getState() == true) {
    wmode =1;
    _root.freset();
    //
    genset3._visible = false;
    genset5._visible = false;
    genset4._visible = true;
    graphicsourcebut._visible = false;
    graphicrebut._visible = false;
    graphicraybut._visible = true;
    detailhide._visible = true;
    //
    anaquality.setEnabled(false);
    butgraphic.gotoAndStop(2);
    reftimeslide.setEnabled(false);
    reftimegraphic.gotoAndStop(2);
    //
    direct._visible = true;
    ld._visible = true;
    node._visible = true;
    status._visible = true;
    removeg._visible = false;
    //
    _root.clearmeter();
    _root.fmeterhide();
    //
    loopslide.setValue(10);
```

```
  allangslide.setValue(30);
  initdbslide.setValue(100);
  dbfarslide.setValue(1);
  //
  startbut._visible = true;
  stopbut._visible = true;
  cancelbut._visible = false;
  //
  envigraphic._visible = true;
  airtempslide._visible = true;
  //
  genset4.setState(true);
  _root.fgenset();
  //
  //
}
if (workmode2.getState() == true||workmode3.getState() == true||workmode4.getState() == true) {
  if (workmode2.getState() == true) {
    wmode = 2;
    startbut._visible = true;
    stopbut._visible = false;
    cancelbut._visible = true;
    hidedb._visible = false;
    hideecho._visible = true;
    removeg._visible = true;
    reftimeslide.setEnabled(true);
    reftimegraphic.gotoAndStop(1);
  }
  if (workmode3.getState() == true) {
    wmode = 3;
    startbut._visible = true;
    stopbut._visible = false;
    cancelbut._visible = true;
    hidedb._visible = false;
    hideecho._visible = false;
    removeg._visible = true;
    reftimeslide.setEnabled(true);
    reftimegraphic.gotoAndStop(1);
  }
  if (workmode4.getState() == true) {
    wmode = 4;
    startbut._visible = true;
    stopbut._visible = true;
    cancelbut._visible = false;
    hidedb._visible = true;
    hideecho._visible = true;
    removeg._visible = false;
    reftimeslide.setEnabled(false);
    reftimegraphic.gotoAndStop(2);
  }
  _root.freset();
  //
  genset3._visible = true;
  genset4._visible = false;
  genset5._visible = true;
```

```
    graphicsourcebut._visible = true;
    graphicraybut._visible = false;
    graphicrebut._visible = true;
    detailhide._visible = false;
    //
    anaquality.setEnabled(true);
    butgraphic.gotoAndStop(1);
    //
    direct._visible = false;
    Id._visible = false;
    node._visible = true;
    status._visible = true;
    //
    _root.fmetershow();
    _root.fmeternum();
    //
    allangslide.setValue(360);
    loopslide.setValue(45*quality);
    //
    envigraphic._visible = true;
    airtempslide._visible = true;
    //
    genset5.setState(true);
    _root.fgenset();
  }
}
//
//
function fgenset() {
  if (genset1.getState() == true) {
    _root.clearmatpoint();
    _root.clearmeter();
    _root.hideeverything();
    _root.showroom();
  }
  if (genset2.getState() == true) {
    _root.clearmeter();
    _root.hideeverything();
    _root.showmat();
  }
  if (genset3.getState() == true) {
    _root.clearmeter();
    _root.clearmatpoint();
    _root.hideeverything();
    _root.showsource();
  }
  if (genset4.getState() == true) {
    _root.clearmeter();
    _root.clearmatpoint();
    _root.hideeverything();
    _root.showray();
  }
  if (genset5.getState() == true) {
    _root.clearmatpoint();
    _root.hideeverything();
```

```
        _root.showre();
    }
  }
 function hideeverything() {
    graphicroom._visible = false;
    roomlist._visible = false;
    controlnum._visible = false;
    symoff._visible = false;
    sym1._visible = false;
    sym2._visible = false;
    sym3._visible = false;
    //
    graphicmat._visible = false;
    matlist._visible = false;
    matadd._visible = false;
    absorbslide._visible = false;
    flrabslide._visible = false;
    cilabslide._visible = false;
    //
    graphicsource._visible = false;
    dbsamplelist._visible = false;
    dbfarslide._visible = false;
    initdbslide._visible = false;
    //
    graphicray._visible = false;
    loopslide._visible = false;
    allangslide._visible = false;
    //
    graphicre._visible = false;
    meternumslide._visible = false;
    meterdistance._visible = false;
    hitareaslide._visible = false;
    metmodeoff._visible = false;
    meterangleslide._visible = false;
    metmoderad._visible = false;
 }
 function showroom() {
    graphicroom._visible = true;
    roomlist._visible = true;
    controlnum._visible = true;
    symoff._visible = true;
    sym1._visible = true;
    sym2._visible = true;
    sym3._visible = true;
 }
 function showmat() {
    graphicmat._visible = true;
    matlist._visible = true;
    matadd._visible = true;
    absorbslide._visible = true;
    flrabslide._visible = true;
    cilabslide._visible = true;
 }
 function showsource() {
    graphicsource._visible = true;
    dbsamplelist._visible = true;
    dbfarslide._visible = true;
    initdbslide._visible = true;
}
function showray() {
    graphicray._visible = true;
    loopslide._visible = true;
    allangslide._visible = true;
}
function showre() {
    graphicre._visible = true;
    meternumslide._visible = true;
    meterdistance._visible = true;
    hitareaslide._visible = true;
    metmodeoff._visible = true;
    meterangleslide._visible = true;
    metmoderad._visible = true;
}
function froomlist() {
    roomsample = roomlist.getSelectedItem().data;
    if (roomsample == 0) {
    }
    if (roomsample == 1) {
        roomsampleindex = 1;
        controlnum.setValue(4);
        symoff.setState(true);
        roomsampleindex = 0;
    }
    if (roomsample == 2) {
        roomsampleindex = 1;
        controlnum.setValue(6);
        roomsampleindex = 0;
    }
    if (roomsample == 3) {
        roomsampleindex = 1;
        controlnum.setValue(8);
        roomsampleindex = 0;
    }
```

ประวัติผู้เขียนวิทยานิพนธ์

นาย กฤตภัค กุลบศย์ เกิดเมื่อวันที่ 22 มิถุนายน พ.ศ.2522 กรุงเทพมหานคร สำเร็จการศึกษาในระดับปริญญาตรี สถาปัตยกรรมศาสตร์บัณฑิต(สถ.บ.เกียรตินิยมอันดับสอง) จากคณะสถาปัตยกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2543 และเข้าศึกษาในระดับปริญญาโท ในสาขาคอมพิวเตอร์ในงานออกแบบสถาปัตยกรรม ภาควิชาสถาปัตยกรรม คณะสถาปัตยกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2544 โดยทุนโครงการพัฒนาอาจารย์ของทบวงมหาวิทยาลัย