

SDT
510.112
약225표
=2

교육학석사 학위논문

플래시 무비를 이용한 동적 차트의 제작

지도교수 송 태 성

2003년 8월

부산대학교 교육대학원

수학교육전공

곽 동 옥

648995

곽동옥의 교육학석사 학위 논문을 인준함

2003년 6월 일

심사위원장 박 재 결 (인)

심 사 위 원 김 부 윤 (인)

심 사 위 원 송 태 성 (인)

목 차

1. 서론	1
2. 파이 차트	4
3. 바 차트	19
참고문헌	41
요약(영문)	42

1. 서론

매크로미디어 플래시(Macromedia Flash)는 인터넷 멀티미디어 저작 도구이다. 플래시를 사용하여 풍부한 인터넷 콘텐츠와 응용 프로그램을 신속하게 개발할 수 있다. 플래시 무비 (Flash Movies)는 웹에서 사용되는 그래픽, 텍스트, 애니메이션, 대화식 무비 등으로 구성된다. 플래시 무비는 기본적으로 벡터 그래픽으로 구성되나 비트맵 그래픽, 사운드, 비디오 등으로 포함할 수 있다.

매크로미디어 플래시는 상호작용을 위한 강력하고 확장 가능한 객체 모델을 제공한다. 이 객체 모델에는 액션 스크립트(Action Script), 프로토타입 기반 상속, 이벤트 기반 프로그래밍, 도형 드로잉 API 등이 포함된다. 풍부한 클라이언트 기술의 특성, 매크로미디어 플래시가 제공하는 클라이언트 기술, 매크로미디어 플래시 기술의 응용 등은[3]과[8]에 자세히 소개되어 있다. 플래시에 내장된 액션 스크립트(ActionScript)는 프로그래밍과 스크립팅을 위한 객체 모델이다. 액션 스크립트는 웹 브라우저가 해석할 수 있는 스크립트 언어로 넷스케이프에서 개발한 자바 스크립트(JavaScript)와 마이크로소프트에서 개발한 Jscript와 본질적으로 같은 언어이다. 웹 스크립트 언어의 구문(syntax)이 자바의 구문과 비슷하지만 웹 스크립트 언어는 본질적으로 다른 언어이다. 웹 스크립트 언어에는 Java의 핵심 개념인 class 개념이 포함되어 있지 않다. 또한, Java 는 변수의 타입을 명확하게 선언해야 되는 언어이지만 웹 스크립트는 변수의 타입을 명확하게 선언할 필요가 없는 언어이다. 자바 스크립트(Java Script)와 액션 스크립트 언어에 내장되어 있는 객체(object), 메서드(method), 속성(property), 액션 스크립트 언어의 연산 및 구문 등은 [5]에 취급되어 있고 [9]에는 액션 스크립트 코딩 표준이 취급되어 있다. 자바 스크립트 언어의 구조는[7]에 자세히 취급되어 있다.

플래시 무비를 사용하여 중등학교나 대학교의 수학교육에서 유용하게 사

용할 수 있는 [4]와 [6]에 다양한 대화식 정적 무비와 동적 무비를 제작할 수 있다. 대화식 정적 무비와 동적 무비를 제작하려면 먼저 무비의 기본 구조를 디자인 하고 무비에 사용할 심벌(그래픽, 버튼, 무비클립)을 제작하여야 한다. 무비의 그래픽 화면을 구성한 후에 대화식 동적 무비가 되도록 프레임이나 심벌의 인스턴스에 스크립트 코드를 삽입해야 한다. [2]에는 수학교육에 활용할 수 있는 플래시 무비의 다양한 기능이 소개되어 있고 수학교육에서 유용하게 사용할 수 있는 다양한 HTML Form 계산기와 Frash Form 계산기 제작 방법이[1]에 취급되어 있다. 플래시 무비는 기본적으로 벡터그래픽으로 구성되나 비트맵 그래픽, 사운드, 비디오 등을 포함할 수 있다.

플래시 파일에는 플래시 문서(Frash documents)와 플래시 무비(Frash movies)가 있다. 플래시 무비는 확장자가 FLA인 파일이고 플래시 무비는 확장자가 SWF인 파일이다. 플래시 플레이어(Flash Player)가 재생할 수 있는 파일은 플래시 문서가 아니고 플래시 무비이다. 플래시에서 확장자가 FLA인 플래시 문서만 편집할 수 있고 확장자가 SWF인 플래시 무비는 편집할 수 없다. 확장자가 SWF인 플래시 무비는 플래시에 내장된 플래시 플레이어나 HTML문서에 삽입된 웹 브라우저에서 실행되는 파일이다. 플래시로 제작한 플래시 문서를 웹 브라우저에서 실행되는 플래시 무비로 만들려면 File-Publish 메뉴를 선택하여 FLA 파일을 SWF 파일로 해야 한다. 그러면 플래시 무비와 플래시 무비를 Object tag로 연결시킨 HTML 파일이 생성된다.

뉴스나 신문, 선거 날에 투표결과를 보여줄 때 가장 자주 사용되는 형태의 그래프가 파이 차트나 바 차트 이다. 그 이름에서 알 수 있듯이 사용자가 지정하는 데이터를 바나 파이조각 형태로 나타내주는데, 얼마만큼 지분을 차지하는지를 한눈에 가장 잘 나타내어줄 수 있는 형태의 그래프라고 할 수 있다. 사용자가 데이터를 선택하여 파이 차트를 그리게 되면 그 데이터의 모든 값을 합한 값으로 전체 파이를 만들고, 이들의 각각의 데이터가 치

지하는 비율을 사용해서 각 데이터에 대한 조각으로 나누게 된다.


플래시의 그래픽의 기능과 플래시에 내장된 액션 스크립트 언어를 사용하여 대화식 정적 무비와 대화식 동적 애니메이션을 제작하는 방법을 소개하고 통계의 지도는 생활 속에서 만나는 자료를 효율적으로 정리, 분석함으로써 유용한 정보를 얻는데 효과적인 도구가 통계적인 방법임을 알 수 있게 현실적인 과제를 다루며, 창의적인 문제해결에 적용할 수 있도록 실제적이면서 통합적인 지도를 한다.

제1절에서는 플래시의 그래픽 기능과 플래시에 내장된 액션스크립트 언어를 사용하여 두 개의 파이 차트 무비(PieChart1.fla 무비, PieChart2.fla 무비)를 제작하고 이들 차트를 사용하여 2000년도 우리나라의 연령별 총인구 분포를 분석한다. PieChart1 무비는 대화식 정적 무비이고 PieChart2 무비는 대화식 동적 애니메이션 무비이다.

제2절에서는 하나의 대화식 정적 무비(BarChart1.fla 무비)와 두 개의 대화식 동적 무비(BarChart2.fla 무비, BarChart3.fla 무비)를 제작한다. BarChart2는 대화식 동적 애니메이션 무비이고 BarChart3는 여러 영역의 두 자료를 한눈으로 쉽게 비교할 수 있는 대화식 동적 애니메이션 무비이다. BarChart1 무비와 BarChart2 무비를 사용하여 2002년도 도시 전가구 소비지출을 분석하고 BarChart3 무비를 사용하여 2003학년도 대학수학능력시험 결과를 분석한다.

1. 파이 차트

이 절에서는 플래시의 그래픽 기능과 플래시에 내장된 액션스크립트 언어를 사용하여 두 개의 파이 차트 무비(PieChart1.fla 무비, PieChart2.fla 무비)를 제작한다. PieChart1 무비는 대화식 정적 무비이고 PieChart2 무비는 대화식 동적 애니메이션 무비이다. 아래 그림은 두 개의 파이 차트 무비의 1 프레임에 삽입한 객체의 그래픽이다. 3개의 Input Text 필드의 변수 이름을 각각 title, box1, box2로 지정한다.

Title	<input type="text"/>	
Label	<input type="text"/>	
Data	<input type="text"/>	

두 개의 파이 차트 무비 제작에 3개의 배열(Color 배열, Label 배열, 그리고 Data 배열)을 사용한다. Color 배열은 1 프레임에 삽입한다. 변수 이름이 box1, box2인 Input Text 필드에 콤마로 분리하여 입력한 자료를 배열로 바꾸어 주는 스크립트 코드를 버튼 인스턴스에 삽입한다.

1 프레임에 삽입한 `colorString.split(",");` 메서드는 스트링을 배열로 바꾸어주는 명령이고 `parseInt(colorHex[i], 16);` 메서드는 16진수를 10진수로 바꾸는 명령이다. PieChart1 무비는 정적 무비이므로 `gotoAndStop (2);` 명령을 버튼 인스턴스에 삽입하고 PieChart2 무비는 애니메이션 무비이므로 이 명령 대신에 `gotoAndPlay (2);` 명령을 버튼 인스턴스에 삽입한다. 1 프레임과 버튼 인스턴스에 삽입한 스크립트 코드는 다음과 같다.

[1 프레임]

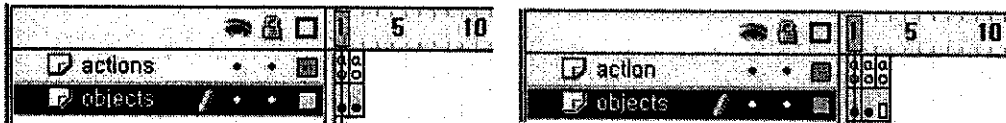
```
stop ();  
colorString= "ff0000, 00ff00, ffff00, 00ffff, 990000, 009900, 99cc66,  
ffffff, ffccff,000099, 660099, ccff00, ffcc00, 00ffcc, d2db28";  
colorHex = colorString.split(",");  
for (var i in colorHex) {  
    colorHex[i] = parseInt(colorHex[i], 16);  
}
```

[버튼] 인스턴스

```
on (release) {  
    labelString= eval ("box1");  
    dataString = eval ("box2");  
    label = labelString.split(",");  
    data= dataString.split(",");  
    for (var i in data) {  
        data[i] = Number(data[i]);  
    }  
    n = data.length;  
    myColor= colorHex.slice(0,n);  
    gotoAndStop (2);  
}
```

아래 그림과 같이 PieChart1 무비의 Timeline은 2개의 프레임으로 구성하고 PieChart2 무비의 Timeline은 3개의 프레임으로 구성한다. 앞에서 언급한 바와 같이 두 무비의 1 프레임에 3개의 Input Text 필드와 하나의 버튼 인스턴스, 그리고 적당한 텍스트를 삽입한다. PieChart1 무비의 2 프레

임과 PieChart2 무비의 2 프레임에 변수 이름이 chartTitle인 적당한 크기의 Dynamic Text 필드와 적당한 텍스트를 삽입한다.



PieChart1 무비의 파이 조각은 beginFill, moveTo, lineTo, endFill 등의 명령을 사용하여 제작하고 PieChart2 무비의 파이 조각은 복제 명령과 마스크 명령을 사용하여 제작한다.

무비 클립의 복제 명령에는 attachMovie 명령과 duplicateMovieClip 명령이 있다. attachMovie 명령으로 무비 클립 인스턴스를 복제하려면 원본 무비 클립 심벌에 Linkage를 설정하여 무비 클립 심벌의 Identifier를 지정해야 하고 duplicateMovieClip 명령을 사용하여 복제하려면 원본 무비 클립 심벌의 인스턴스를 작업 영역(스테이지의 외부)에 삽입하고 인스턴스의 이름을 지정해야 한다.

먼저 PieChart1 무비를 제작한다. PieChart1 무비 제작에 다음 2개의 심벌을 사용한다: draw, pieText. 여기서 draw는 버튼 심벌이고 pieText는 파이 조각의 레이블을 표현할 무비 클립 심벌이다. pieText는 변수 이름이 각각 box3, box4인 2개의 Dynamic Text 필드로 구성된 무비 클립이다. 무비 클립 pieText에 Linkage를 설정하고 Identifier를 pieText로 지정한다.

2 프레임에 데이터의 합, 파이 조각의 전체에 대한 비율(%) 등을 계산할 수 있는 스크립트 코드와 파이 조각의 중심각 생성 코드, 파이 조각의 색 지정 코드, 파이 조각 생성 코드, 레이블의 복제 코드, 레이블의 위치 지정

코드 등을 삽입하면 PieChart1 무비의 제작이 완료된다. 다음의 스크립트 코드는 2 프레임에 삽입한 스크립트 코드이다.

```
stop();
chartTitle = title;
radius = 150; x0=400; y0=280;
for (var i=0; i<n; i++){dataSum += data[i];}
percent = new Array ();
for (i=0; i < n; i++) {
    percent[i] = Math.round((data[i]/dataSum)*1000)/10;
}

angFrom=0; angTo=0;
for (var i=0;i<n;i++){
    angTo = angFrom + ((data[i]/dataSum)*360)
    _root.createEmptyMovieClip ("wedge" + i, 1);
    with (_root.wedge + i){
        beginFill (myColor[i], 100);
        lineStyle (2, 0x000000, 100);
        moveTo (x0, y0);
        for (var k=angFrom-1;k<angTo;k++){
            lineTo (radius*Math.cos(Math.PI/180 * k)+ x0,
                    radius*Math.sin(Math.PI/180 * k)+ y0);
        }
        lineTo (x0, y0);
        endFill();
    }
}
```

```

var bisect = angFrom + (angTo-angFrom)/2;
_root.createEmptyMovieClip ("line" + i, i+ 30);
with (_root.line + i){
    beginFill (0xffffffff, 0);
    lineStyle (1, 0x000000, 100);
    moveTo (radius*.7*Math.cos(Math.PI/180 * bisect)+ x0,
            radius*.7*Math.sin(Math.PI/180 * bisect)+ y0);
   .lineTo (radius*1.1*Math.cos(Math.PI/180 * bisect)+ x0,
            radius*1.1*Math.sin(Math.PI/180 * bisect)+ y0);
    endFill();
}

_root.attachMovie("pieText", "pieText"+i, i+ 10)
sliceName = eval("pieText"+ i);
sliceName.box3 = label[i];
sliceName.box4 = data[i] + " (" + percent[i] + " %)";
radius1 = radius *1.3
sliceName._x = radius1 * Math.cos(Math.PI/180 * bisect)+ x0;
sliceName._y = radius1 * Math.sin(Math.PI/180 * bisect)+ y0;
angFrom=angTo
}

```

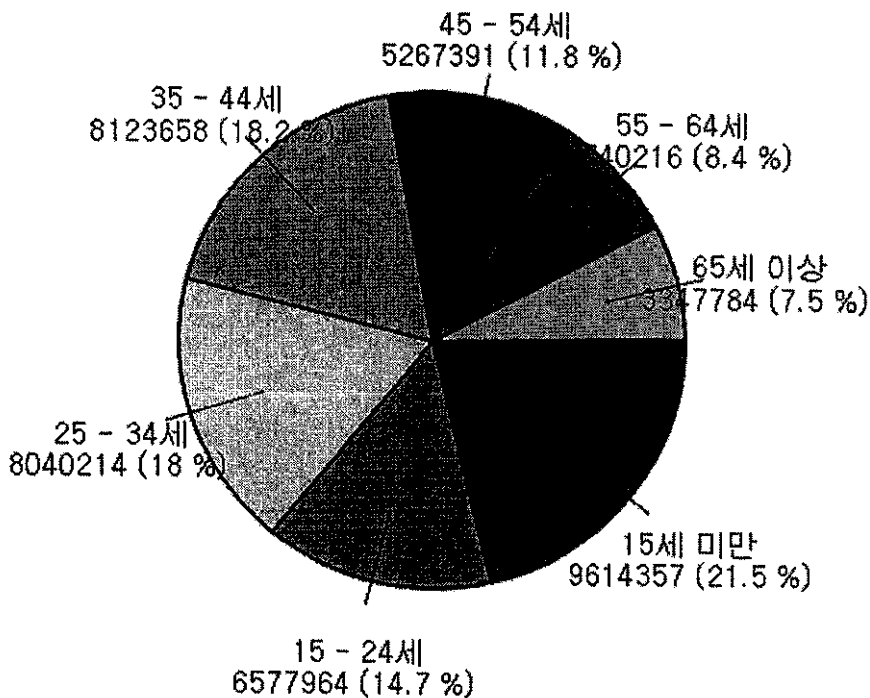
다음 데이터는 2000년도 우리나라의 연령별 총인구 분포 표이다(출처: 통계청). 이 데이터의 분석 결과를 PieChart1 무비로 표현하려면 Title 필드에 제목을 입력하고 Label 필드와 Data 필드에 각각 연령과 인구를 콤마로 분리하여 입력한 후에 DrawPie 버튼을 누르면 된다.

연령	인구	연령	인구
15세 미만	9,614,357	45 - 54세	5,267,391
15 - 24세	6,577,964	55 - 64세	3,740,216
25 - 34세	8,040,214	65세 이상	3,347,784
35 - 44세	8,123,658	합계	44,711,584

(65세 이상에 연령 미상 963명 포함)

아래 그림은 2000년도 우리나라의 연령별 총인구 분포를 PieChart1 무비로 표현한 그래픽이다.

연령별 총인구 (2000년도)



PieChart2 무비를 제작한다. PieChart2 무비는 대화식 동적 애니메이션 무비이다. PieChart2 무비 제작에 다음 6개의 심벌을 사용한다.

- Button 심벌: button, draw
- Movie Clip 심벌: bclip, disk3d, dummy, pie3d

심벌 이름이 button인 버튼 심벌은 Up 프레임에 Blank Keyframe을 삽입하고 나머지의 프레임에 작은 fill 원을 삽입한 버튼으로 bclip 무비 클립에 삽입할 심벌이다. disk3d 무비 클립은 타원형 3d fill 원판으로 pie3d 무비 클립에 삽입할 심벌이고 dummy 무비 클립은 pie3d 무비 클립에 삽입할 빈 심벌이다. 2개의 무비 클립(bclip와 pie3d 무비 클립)에 Linkage를 설정하고 Identifier를 각각 bclip와 pie3d로 지정한다.

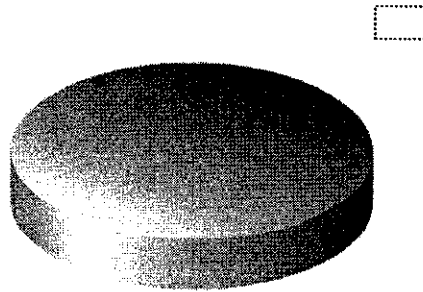
파이 조각의 레이블 무비 클립인 bclip의 그래픽 화면은 아래 그림과 같이 버튼, 3d fill 원판, 그리고 Dynamic Text 필드로 구성한다(버튼 심벌은 3d fill 원판 아래에 삽입한다).



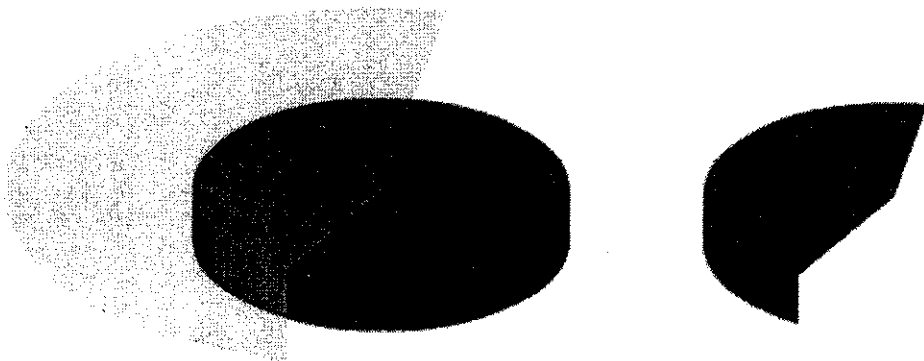
레이블 무비 클립인 bclip에 삽입한 텍스트 필드의 인스턴스 이름을 txt로 지정하고 다음의 스크립트 코드를 bclip 무비 클립의 1 프레임에 삽입한다. pieArray 배열과 bclip의 id 속성을 정의하는 스크립트 코드는 메인 무비의 2 프레임에 삽입한다.

```
txt.autoSize=true;
button.onRollOver = function() {_root.pieArray[id].status = "on";};
button.onRollOut = function () {_root.pieArray[id].status = "off";};
```

파이 조각 생성 무비 클립인 pie3d의 그래픽 화면은 아래 그림과 같이 disk3d 무비 클립과 Dynamic Text 필드로 구성한다.



파이 조각 생성 무비 클립인 pie3d에 삽입한 disk3d 무비 클립의 인스턴스 이름을 pieBg로 지정하고 Dynamic Text 필드의 인스턴스 이름은 txt로 지정한다. 파이 조각을 만들기 위해서 disk3d 무비 클립을 삽입한 레이어의 아래쪽에 2개의 레이어를 만들고 각 레이어의 1프레임에 dummy 무비 클립(빈 무비 클립)을 삽입하고 인스턴스 이름을 각각 mask와 mcCut로 지정한다. 빈 무비 클립인 mask를 사용하여 마스크를 정의하고 mcCut를 사용하여 파이 조각의 측면을 정의한다. PieChart2 무비에서는 먼저 아래의 왼쪽 그림과 같은 마스크 영역을 만든 후에 setMask 명령을 사용하여 오른쪽 그림과 같은 파이 조각을 생성한다.



파이 조각을 생성하는 스크립트 코드와 파이 조각의 위치를 지정하는 스크립트 코드를 pie3d 무비 클립의 1 프레임에 삽입한다. 다음은 pie3d 무비 클립의 1 프레임에 삽입한 스크립트 코드이다. 아래의 스크립트 코드에서 drawFace 함수는 마스크 영역을 정의하는 함수이고 drawSide 함수는 파이 조각의 측면을 정의하는 함수이다. 그리고 periPoint 함수는 각에 대응하는 타원(3d fill 타원을 구성하는 위쪽 fill 타원의 경계) 위의 점의 위치를 정의하는 함수이다. periPoint 함수와 drawFace 함수의 정의식은 메인 무비의 2 프레임에 삽입한다.

```
txt.autoSize = true;
percentValue = 36;
x0 = this._x; y0 = this._y;
pieBg.setMask(mask);
function show(angleA, angleB) {
    this.angleA = angleA;
    this.angleB = angleB;
    drawFace(mask, angleA, angleB);
    drawSide();
}
function drawSide() {
    mcCut.createEmptyMovieClip("dummy", 1);
    with (mcCut.dummy) {
        thick = 30;
        if (angleA <= 270) {
            pt1 = periPoint(angleA);
            colors = [0xdddddd, 0x999999];
        }
    }
}
```

```

    if (angleB >= 270) {
        pt1 = periPoint(angleB);
        colors = [0x555555, 0xbbbbbb];
    }
    alphas = [100, 100];
    ratios = [0, 0xFF];
    matrix = {matrixType:"box", x:pt1.x, y:0, w:Math.abs(pt1.x),
        h:200, r:0};
    beginGradientFill("linear", colors, alphas, ratios, matrix);
    moveTo(0, 0);
    lineTo(pt1.x, pt1.y);
    lineTo(pt1.x, pt1.y + thick);
    lineTo(0, thick);
    lineTo(0, 0);
    endFill();
}

txt.text = percentValue + "%";
pt = periPoint((angleA + angleB)/2);
txt._x = (pt.x*5/4) - (txt._width)/2;
txt._y = (pt.y*5/4) - (txt._height)/2;
dx = pt.x/10; dy = pt.y/10;
}

this.onEnterFrame = function() {
    if (status == "on") {_x = x0 + dx*4; _y = y0 + dy*4;}
    if (status == "off") {_x = x0; _y = y0;}
};

```


PieChart2 무비의 핵심 구성 요소인 bclip 무비 클립과 pie3d 무비 클립을 제작을 완료했다. 이제 PieChart2 무비가 동적 애니메이션 무비가 되도록 메인 무비의 2 프레임과 3 프레임에 적당한 스크립트 코드를 삽입해야 한다.

2프레임에 pie3d 무비 클립의 복제 코드와 위치 지정 코드, 레이블 무비 클립인 bclip 무비 클립의 복제 코드와 위치 지정 코드, 앞에서 언급한 periPoint 함수와 drawFace 함수의 정의식, bclip의 복제 무비 클립의 id 속성 지정 코드, 무비 클립의 색 지정 함수의 정의식 등을 삽입한다. 색 지정 함수인 mcColor 함수는 8개의 속성(ra, rb, ga, gb, ba, bb, aa, ab)으로 색을 설정하는 setTransform 명령을 사용하여 정의한다. 다음의 코드는 2 프레임에 삽입한 스크립트 코드이다.

```
chartTitle = title; depth = 1;
bcArray = []; pieArray = [];
bcx0 = 550; bcy0 = 90;
piex0 = 250; piey0 = 250;
for (var k = 0; k < n; k++) {
    attachMovie("pie3d", "pie"+k, depth++);
    attachMovie("bclip", "bc"+k, depth++);
    pieArray.push(this["pie"+k]);
    bcArray.push(this["bc"+k]);
    with (this["pie"+k]) {_x = piex0; _y = piey0;}
    with (this["bc"+k]) {
        _y = bcy0 + (350*k/n); _x = bcx0;
        _xscale = 70; _yscale = 70;
    }
}
```

```

    this["bc"+k].id = k;
}
angleA = 30; angleB = 270; radius = 150;
_global.periPoint = function (Angle) {
    var pt = new Object();
    pt.x = radius*Math.cos((Math.PI/180)*Angle);
    pt.y = (0.5)*radius*Math.sin((Math.PI/180)*Angle);
    return pt;
};
_global.drawFace = function (mclip, angleA, angleB) {
    mclip.createEmptyMovieClip("drawer", 1);
    with (mclip.drawer) {
        beginFill(0x00ff00, 100);
        pt1 = periPoint(angleA);
        lineTo(pt1.x, pt1.y);
        if (pt1.y<=0) {lineTo(2*pt1.x, 2*pt1.y);
        } else {lineTo(pt1.x, pt1.y+ 75);}
        var tempAngle = angleA+ 10;
        while (tempAngle<angleB) {
            tempPt = periPoint(tempAngle);
            lineTo(2*tempPt.x, 2*tempPt.y);
            tempAngle += 10;
        }
        pt2 = periPoint(angleB);
        if (pt2.y<=0) {lineTo(2*pt2.x, 2*pt2.y);
        } else {lineTo(pt2.x, pt2.y+ 75);}
        lineTo(pt2.x, pt2.y);
    }
}

```

```

        endFill();
    }
};

function mcColor(mc, rgb) {
    c = new Color(mc);
    cv = rgb;
    r = cv >> 16;
    g = (cv >> 8) & (0xFF);
    b = cv & 0xFF;
    c.setTransform({ra:r*0.6, ga:g*0.6, ba:b*0.6, rb:0, gb:0, bb:0, aa:100,
        ab:0});
}

```

3 프레임에 데이터의 합 생성 코드, 데이터의 크기가 가장 큰 원소의 인덱스 계산 코드, 파이 조각의 위치 지정 코드, 파이 조각의 전체에 대한 비율(%) 계산 코드, 파이 조각의 중심각 크기 지정 코드, 파이 조각의 색 지정 코드, 레이블 무비 클립에 삽입할 텍스트 지정 코드, 레이블 무비 클립의 색 지정 코드 등을 삽입하면 PieChart2 무비의 제작이 완료된다. 파이 조각의 위치는 크기가 가장 큰 원소에 대응하는 파이 조각을 90°를 출발점으로 하여 배치한다. 3 프레임에 삽입한 코드는 다음과 같다.

```

stop();
temp = 0; dataSum = 0;
for (var k = 0; k<n; k++) {dataSum += data[k];}
for (var k = 0; k<n; k++) {
    if (data[k]>temp) {temp = data[k]; maxIndex = k;}
}

```

```

ang0 = 90; index = maxIndex;
for (var k = 0; k<n; k++ ) {
    if (index>= n) {index -= n;}
    value = data[index];
    angOffset = value*360/dataSum;
    var clip2 = pieArray[index];
    with (clip2) {
        percentValue = Math.round(value/dataSum*1000)/10;
        show(ang0, ang0+ angOffset);
        mcColor(pieBg, myColor[index]);
        if (ang0<270) {depth = 10-k;
        } else {depth = 10+ k;}
        swapDepths(depth);
    }
    ang0 += angOffset;
    index++;
}

for (var k = 0; k<n; k++ ) {
    bcArray[k].txt.text = label[k]+ " : "+ data[k];
    mcColor(bcArray[k], myColor[k]);
}

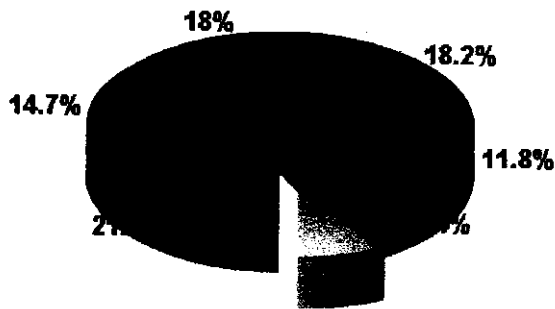
```

PieChart2 무비의 초기 화면에서 3개의 Input Text 필드에 자료를 입력하고 버튼을 누르면 입력한 자료를 그래프로 분석한 애니메이션 파이 차트가 출력된다. 화면에 출력된 파이 차트에서 레이블 버튼 위로 마우스를 가져가면 레이블 버튼에 대응하는 파이 조각이 이동한다. 아래 그림은 2000

년도 우리나라의 연령별 총인구 분포를 PieChart2 무비로 표현한 파이 차트에서 하나의 레이블 버튼 위로 마우스를 가져간 경우의 그래픽이다.

연령별 총인구 (2000년도)

Roll your mouse over a label button

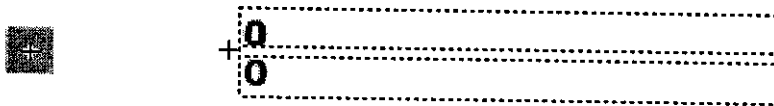


- 15세 미만 : 9614357
- 15 - 24세 : 6577964
- 25 - 34세 : 8040214
- 35 - 44세 : 8123658
- 45 - 54세 : 5267391
- 55 - 64세 : 3740216
- 65세 이상 : 3347784

3. 바 차트

바 차트는 자료의 분포를 수직 바나 수평 바로 표현하는 그래프이다. 바 차트는 주로 특별한 속성을 갖는 범주형 자료의 수나 비율의 그래핑에 사용되는 차트이고 바의 길이는 자료의 수치 값에 대응한다. 이 절에서는 플래시를 사용하여 3개의 바 차트 무비(BarChart1.fla 무비, BarChart2.fla 무비, BarChart3.fla 무비)를 제작한다.

먼저 BarChart1 무비를 제작한다. 이 무비에서는 바를 표현할 fill 사각형 무비 클립 심벌(bar 심벌)과 바의 레이블을 표현할 텍스트 무비 클립 심벌(labelT 심벌), 그리고 하나의 버튼 심벌을 사용한다. 2개의 무비 클립 심벌(bar, labelT)에 Linkage를 설정하고 심벌 이름과 같은 이름으로 Identifier를 지정한다. 아래 그림은 bar 심벌과 labelT 심벌의 그래픽이다.



labelT 무비 클립은 2개의 Dynamic Text 필드로 구성된 무비 클립이다. 2개의 텍스트 필드의 변수 이름을 각각 box3, box4로 지정하고 인스턴스 이름을 각각 tbox3, tbox4로 지정한다. labelT 무비 클립에 삽입한 2개의 텍스트 필드의 크기가 확장될 수 있도록 1 프레임에 다음의 스크립트 코드를 삽입한다: tbox3.autoSize=true; tbox4.autoSize=true;

BarChart1 무비의 Timeline은 2개의 프레임으로 구성한다. 1 프레임에 3개의 Input Text 필드와 버튼 인스턴스를 삽입하고 2 프레임에 차트의 제목을 출력할 Dynamic Text 필드를 삽입한다. 4개의 텍스트 필드의 변수 이름을 각각 title, box1, box2, chartTitle로 지정한다. 아래 그림은 1 프레임에 삽입한 객체의 그래픽이다.

Title

Label



Data

BarChart1 무비 제작에 사용할 심벌 제작과 BarChart1 무비의 그래픽 화면이 완성되었다. 이제 BarChart1 무비가 대화식 무비가 되도록 프레임과 버튼 인스턴스에 스크립트 코드를 삽입한다. BarChart1 무비에서도 앞에서 제작한 두 개의 파이 차트 무비에서와 같이 stop(); 액션과 Color 배열을 1 프레임에 삽입하고 Input Text 필드에 콤마로 분리하여 입력한 레이블 자료와 레이블에 대응하는 수치 자료를 배열로 바꾸어 주는 스크립트 코드를 버튼 인스턴스에 삽입한다.

2 프레임에 데이터의 합 및 백분을 계산 코드, 데이터의 최대값 계산 코드, 바의 복제 코드, 바의 높이 및 데이터 값에 대응하는 바의 폭 지정 코드, 바의 색 지정 코드, 레이블 복제 코드, 레이블 텍스트 지정 코드, 레이블의 위치 지정 코드, 수직 선분 그리기 코드 등을 삽입하면 BarChart1 무비의 제작이 완료된다. BarChart1 무비에서 데이터 값에 대응하는 바는 수평 바이다. 다음의 스크립트 코드는 2 프레임에 삽입한 코드이다.

```
chartTitle = title;
for (var i=0; i < n; i++) {dataSum += data[i];}
percent = new Array ();
for (var i=0; i < n; i++) {
```

```

    percent[i] = Math.round((data[i]/dataSum)*1000)/10;
}
max = 0;
for (var i=0; i<n; i++) {max = Math.max(max, data[i]);}
depth = 1;
for (var i=0; i<n; i++) {
    _root.attachMovie("bar", "bar"+i, depth++);
    var bari = "bar"+i;
    var v1 = _root[bari]._height = 450/n;
    var v2 = _root[bari]._width= data[i]/max*550;
    var v3 = v1*2/3;
    _root[bari]._height = v3;
    _root[bari]._y = 70 + v1*i + v3/2;
    _root[bari]._x = 50 + v2/2 ;
    barColor = new Color(bari);
    barColor.setRGB(myColor[i]);
    _root.attachMovie ("labelT", "labelT"+i, depth++);
    dlabel = eval("labelT"+i);
    dlabel.box3 = label[i];
    dlabel.box4 = data[i] + "(" + percent[i] + " %)";
    dlabel._x = 50 + v2;
    dlabel._y = 70 + v1*i + v3/2;
}
_root.createEmptyMovieClip("line", depth++);
with(_root.line) {
    lineStyle(2, 0x000000,100);
    moveTo(50,50);
}

```



```
lineTo(50, 520);
```

```
}
```

다음 데이터는 2002년도 우리나라의 도시 전가구 분기별(평균) 소비지출 분포표이다(출처: 통계청 도시가계 조사).

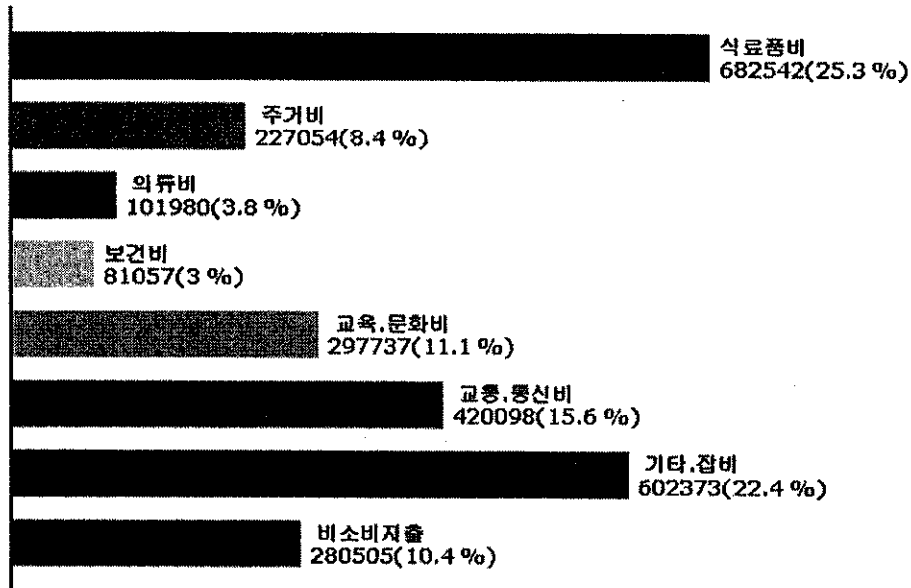
이 데이터의 분석 결과를 BarChart1 무비로 표현하려면 Title 필드에 제목을 입력하고 Label 필드와 Data 필드에 각각 품목과 지출액을 콤마로 분리하여 입력한 후에 drawBar 버튼을 누르면 된다.

2002년도 도시 전가구 소비지출 분포표(분기별 평균 금액)

가계지출	지출금액(원)	가계지출	지출금액(원)
식료품비	682,542	교육, 문화비	297,737
주거비	227,054	교통, 통신비	420,098
의류비	101,980	기타, 잡비	602,373
보건비	81,057	비소비지출	280,505
합계	2,693,346		

아래 그림은 2002년도 우리나라 도시 전가구 소비지출 분포를 BarChart1 무비를 사용한 자료 분석 그래프이다

도시 전가구 소비지출(2002년)



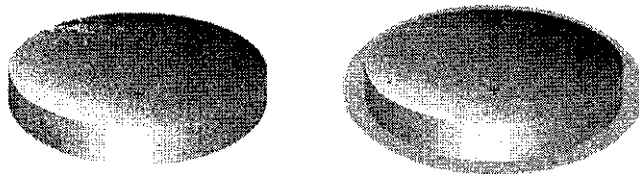
동적 3d 애니메이션 바 차트 무비인 BarChart2 무비를 다음 7개의 심벌을 사용하여 제작한다.

- Button 심벌: button, drawBar
- Movie Clip 심벌: label, slice, text1, text2, text3

심벌 이름이 button인 심벌은 Up 프레임에 Blank Keyframe을 삽입하고 나머지의 프레임에 작은 fill 타원을 삽입한 심벌로 label 무비 클립에 삽입할 버튼 심벌이고 drawBar 심벌은 메인 무비의 1 프레임에 삽입할 버튼 심벌이다. slice 무비 클립은 바 차트의 3d 애니메이션 바를 표현할 무비 클립이고 나머지 4개의 무비 클립(label, text1, text2, text3)은 바의 레이블을 표현할 무비 클립이다. 5개의 무비 클립 심벌에 Linkage를 설정하고 심벌

이름과 같은 이름으로 Identifier를 지정한다.

3d 애니메이션 바를 표현할 slice 무비 클립은 3d fill 타원판이고 label 무비 클립은 button 심벌을 삽입한 3d fill 타원판이다. 아래의 왼쪽 그래픽이 slice 무비 클립 심벌의 그래픽이고 오른쪽 그래픽이 label 무비 클립 심벌의 그래픽이다.



다음의 스크립트 코드는 label 무비 클립의 1 프레임에 삽입한 스크립트 코드로 무비 클립 labelArray[id]의 status를 지정하는 스크립트 코드이다. 아래의 스크립트 코드에서 id는 값이 i로 지정된 label 무비 클립의 복제 무비 클립인 "dlabel"+i의 속성이다. labelArray[i] 무비 클립은 text3 무비 클립의 복제 무비 클립이다.

```
button.onRollOver = function () {_root.labelArray[id].status="on";};  
button.onRollOut = function () {_root.labelArray[id].status="off";};
```

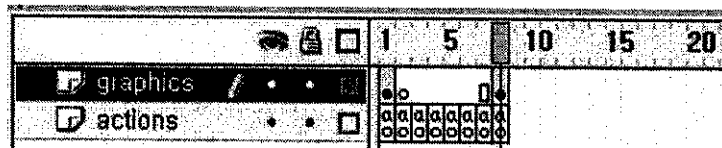
레이블 무비 클립 심벌인 text1, text2, text3 심벌은 Dynamic Text 필드이다. text1 심벌과 text2 심벌에 삽입한 텍스트 필드의 변수 이름을 각각 box3, box4로 지정하고 text3 심벌에 삽입한 텍스트 필드의 변수 이름을 bpx5, 인스턴스 이름을 tbox5로 지정한다. text3 무비 클립의 1 프레임에 다음의 스크립트 코드를 삽입한다. text3 무비 클립에 삽입한 텍스트 필드의 크기와 출력 위치를 지정한 스크립트 코드이다.

```

tbox5.autoSize=true;
this.onEnterFrame = function() {
    if (status == "on") {_x=200; _y =535;}
    if (status == "off") {_x=200; _y = 800;}
};

```

BarChart2 무비의 Timeline은 아래의 그림과 같이 8개의 프레임으로 구성한다.



아래 그림과 같이 1 프레임에 3개의 Input Text 필드, 버튼 인스턴스 등을 삽입한다. 3개의 텍스트 필드의 변수 이름을 각각 title, box1, box2로 지정한다.

Title

Label

Data



Enter values in three input text fields and click the button.

2 프레임에 Blank Keyframe을 삽입하고 7 프레임에 일반 프레임을 삽입한다. 8 프레임에 Keyframe을 삽입하고 차트의 제목을 출력할 텍스트 필드를 삽입한다. 텍스트 필드의 변수 이름을 chartTitle로 지정한다.

BarChart2 무비 제작에 사용할 심벌 제작과 BarChart2 무비의 그래픽 화면이 완성되었다. 이제 BarChart2 무비가 대화식 애니메이션 무비가 되도록 프레임과 버튼 인스턴스에 스크립트 코드를 삽입한다. BarChart2 무비에서도 BarChart1 무비에서와 같이 stop(); 액션과 Color 배열을 1 프레임에 삽입하고 Input Text 필드에 콤마로 분리하여 입력한 레이블 자료와 레이블에 대응하는 수치 자료를 배열로 바꾸어 주는 스크립트 코드를 버튼 인스턴스에 삽입한다.

이제 BarChart2 무비가 애니메이션 무비가 되도록 2 프레임부터 8프레임까지 스크립트 코드를 삽입한다. 2프레임에 데이터의 합과 백분율 계산 코드, 데이터의 최대값 계산 코드, 그리고 데이터의 수치 값에 대응하는 3d 타원판의 개수 계산 코드, 복제 무비 클립의 색 지정 함수 정의 코드 등을 삽입한다. 색 지정 함수인 mcColor 함수는 8개의 속성(ra, rb, ga, gb, ba, bb, aa, ab)으로 색을 설정하는 setTransform 명령을 사용하여 정의한다. 다음의 스크립트 코드는 2 프레임에 삽입한 코드이다.

```
index1 = 0; depth = 1;
for (var i=0; i<n; i++) { dataSum += data[i];}
percent = new Array();
for (var i=0; i<n; i++) {
    percent[i] = Math.round(data[i]/dataSum*1000)/10;
}
dataMax = 0;
for (var i=0; i<n; i++) {
```

```

        if(data[i]>dataMax) {dataMax = data[i];}
    }
    data1 = new Array();
    for (var i=0; i<n; i++) {
        data1[i] = Math.round(data[i]/dataMax*100);
    }
    function mcColor(mc, rgb) {
        c = new Color(mc);
        cv = rgb;
        r = cv >> 16;
        g = (cv >> 8) & (0xFF);
        b = cv & 0xFF;
        c.setTransform({ra:r/2, ga:g/2, ba:b/2, rb:0, gb:0, bb:0, aa:100,
            ab:50});
    }

```

BarChart2 무비가 애니메이션 무비가 되도록 3d 바의 복제 및 색 지정 코드, 3d 바의 폭 및 위치 지정 코드, 3d 바의 위쪽에 출력시킬 수치 레이블의 복제 및 위치 지정 코드 등을 3, 4, 5, 6, 7 프레임에 삽입한다. 3d 바와 수치 레이블은 attachMovie 명령을 사용하여 복제한다. 3, 4, 5, 6, 7 프레임에 삽입한 스크립트 코드는 다음과 같다.

[3 프레임]

```

value1 = data1[index1];
value2 = data[index1];
for (var i =0; i<value1; i++) {
    dsliceA = "dsliceA" + i + "_" + index1;

```

```

    _root.attachMovie ("slice", dsliceA, depth++);
    _root[dsliceA]._x = -150;
    _root[dsliceA]._y = 150;
}
level1= 0;

```

[4 프레임]

```

if (level1 <= value1) {gotoAndPlay (5);
} else {gotoAndPlay (7);}

```

[5 프레임]

```

dsliceA = "dsliceA" + level1 + "_" + index1;
thick = 4*level1;
v1 = 600/n;
v2 = v1*3/4;
xcord = 50 + index1*v1 + v2/2;
mcColor(dsliceA, myColor[index1]);
_root[dsliceA]._width = v2;
_root[dsliceA]._x = xcord;
_root[dsliceA]._y = 490 - thick;
level1++;

```

[6 프레임]

```

gotoAndPlay (4);

```

[7 프레임]

```

if (index1<n) {gotoAndPlay (3);

```

```

} else {gotoAndStop (8);}
dtextA = "text1" + level1 + index1;
_root.attachMovie("text1", dtextA, depth++);
thick = 4*level1;
v1 = 600/n;
v2 = v1*3/4;
xcord = 50 + index1*v1 + v2/2;
_root[dtextA]._width = v2;
_root[dtextA]._x = xcord;
_root[dtextA]._y = 470 - thick;
_root[dtextA].box3 = value2;
index1++;

```

8 프레임에 차트의 제목 출력 코드, 레이블 버튼의 복제 및 색 지정 코드, 레이블 버튼의 크기 및 위치 지정 코드, 2개의 레이블 텍스트의 복제 및 위치 지정 코드, 그리고 복제 레이블 버튼의 id 속성 지정 코드 등을 삽입하면 BarChart2 무비의 제작이 완료된다. 아래의 스크립트 코드는 8 프레임에 삽입한 코드이다.

```

chartTitle = title;
labelArray = new Array();
for (var i = 0; i < n; i++) {
    dlabel = "dlabel" + i;
    _root.attachMovie ( "label", dlabel, depth++ );
    mcColor(dlabel, myColor[i]);
    _root[dlabel]._width = 50;
    _root[dlabel]._x = 700;

```



```

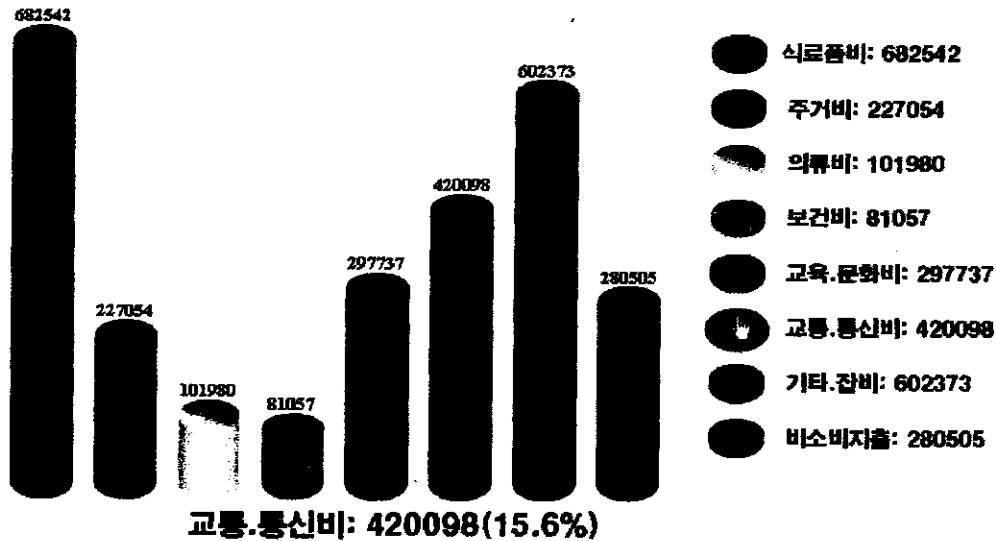
_root[dlabel]._y = 100 + 50 * i;
dtextB = "dtextB" + i;
_root.attachMovie("text2", dtextB, depth++);
_root[dtextB].box4 = label[i] + ": " + data[i];
_root[dtextB]._x = 735;
_root[dtextB]._y = 100 + 50 * i;
_root["dlabel" + i].id = i;
_root.attachMovie("text3", "text3" + i, depth++);
var text3i = _root["text3" + i];
labelArray.push(text3i);
text3i.box5 = label[i] + ": " + data[i] + "(" + percent[i] + "%)";
text3i._x = 0;
text3i._y = 800;
}

```

BarChart2 무비의 초기 화면에서 3개의 Input Text 필드에 자료를 입력하고 버튼을 누르면 입력한 자료를 그래프로 분석한 애니메이션 바 차트가 출력된다. 화면에 출력된 바 차트에서 레이블 버튼 위로 마우스를 가져가면 레이블 버튼에 대응하는 바 조각이 이동한다.

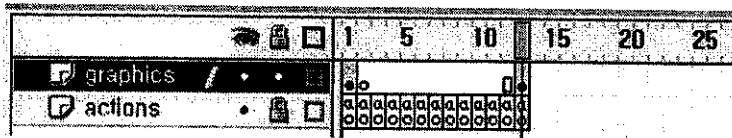
아래 그림은 2002년도 우리나라의 도시 전가구 소비지출을 BarChart2 무비를 사용한 자료 분석의 바 차트에서 하나의 레이블 버튼 위로 마우스를 가져간 경우의 그래픽이다.

도시 전가구 소비지출 (2002년)




2개의 그룹의 범주형 자료의 각각을 3d 애니메이션 바로 표현할 수 있는 무비인 BarChart3 무비를 제작한다. BarChart3 무비에서는 바를 표현할 2개의 3d 사각형판 무비 클립 심벌(slice1, slice2), 바의 레이블을 표현할 2개의 Dynamic Text 무비 클립 심벌(text1, text2), 그리고 하나의 버튼 심벌을 사용한다. 2개의 텍스트 무비 클립에 삽입한 텍스트 필드의 변수 이름을 각각 field1, field2로 지정한다. 4개의 무비 클립 심벌에 Linkage를 설정하고 심벌 이름과 같은 이름으로 Identifier를 지정한다.

BarChart3 무비의 3d 애니메이션 바는 attachMovie 명령을 사용하여 제작한다. BarChart3 무비의 Timeline은 아래의 그림과 같이 18개의 프레임으로 구성한다.

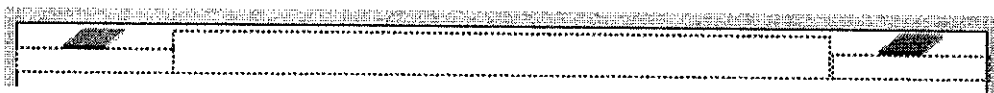


아래 그림과 같이 1 프레임에 5개의 Input Text 필드, 버튼 인스턴스 등을 삽입한다. 5개의 텍스트 필드의 변수 이름을 각각 title, group, box1, box2, box3로 지정한다.

Title
Group
Label
Data1
Data2



2 프레임에 Blank Keyframe을 삽입하고 12 프레임에 일반 프레임을 삽입한다. 13 프레임에 Keyframe을 삽입하고 차트의 제목을 출력할 적당한 크기의 Dynamic Text 필드와 그룹 레이블을 출력할 객체의 그래픽을 스테이지의 상단에 삽입한다. 아래의 그래픽은 13 프레임에 삽입한 객체의 그래픽이다. 차트의 제목을 출력할 텍스트 필드의 변수 이름을 chartTitle로 지정하고 2개의 그룹 레이블에 삽입한 Dynamic Text 필드의 변수 이름을 각각 group0과 group1로 지정한다.



1 프레임에 stop(); 액션을 지정한다. BarChart1 무비에서와 같이 텍스트

필드에 콤마로 분리하여 입력한 레이블 자료와 레이블에 대응하는 수치 자료를 배열로 바꾸어 주는 스크립트 코드를 버튼 인스턴스에 삽입한다. 아래의 스크립트 코드는 버튼 인스턴스에 삽입한 코드이다.

```
on (release, keyPress "<Enter>") {
    groupString= eval ("group");
    labelString= eval ("box1");
    data1String = eval ("box2");
    data2String = eval ("box3");
    group = groupString.split(",");
    label = labelString.split(",");
    data1= data1String.split(",");
    data2= data2String.split(",");
    for (var i in data1) {data1[i] = Number(data1[i]);}
    for (var i in data2) {data2[i] = Number(data2[i]);}
    n = data1.length;
    data = data1.concat(data2);
    gotoAndPlay (2);
}
```

이제 BarChart3 무비가 애니메이션 무비가 되도록 2 프레임부터 13프레임까지 스크립트 코드를 삽입한다. 버튼 인스턴스에서 텍스트 필드에 입력한 2개의 수치 데이터의 배열은 각각 data1, data2로 정의하고 이들 2개의 데이터를 연결한 데이터의 배열을 data로 정의했다. 2 프레임에서 data 배열의 최대값을 기준으로 하여 새로운 2개의 수치 데이터 배열(data1A 배열과 data2A 배열)을 정의한다. 이들 2개의 수치 데이터 배열은 각각 data1 배열과 data2 배열에 대응하는 배열이다. 다음의 스크립트 코드는 2 프레임

에 삽입한 코드이다.

```
index1 = 0; index2 = 0; depth = 1;
dataMax = 0;
for (var i=0; i<data.length; i++) {
    if(data[i]>dataMax) {dataMax = data[i];}
}
data1A = new Array();
for (var i=0; i<n; i++) {
    data1A[i] = Math.round(data1[i]/dataMax*80);
}
data2A = new Array();
for (var i=0; i<n; i++) {
    data2A[i] = Math.round(data2[i]/dataMax*80);
}
```

버튼 인스턴스에 삽입한 스크립트 코드에 의해서 그룹 레이블의 이름은 group[0]과 group[1]이다. 3, 4, 5, 6, 7 프레임에 group[0] 자료의 3d 애니메이션 바의 복제 및 위치 지정 코드, 수치 레이블의 복제 및 위치 지정 코드 등을 삽입한다.

[3 프레임]

```
value1 = data1A[index1];
value2= data1[index1];
for (var i=0; i<value1; i++) {
    dsliceA= "dsliceA" + i + index1;
    _root.attachMovie ("slice1", dsliceA, depth++);
    _root[dsliceA]._x = -150;
```

```

    _root[dsliceA]._y = 150;
}
level1= 0;

```

[4 프레임]

```

if (level1 <= value1) {gotoAndPlay (5);
} else {gotoAndPlay (7);}

```

[5 프레임]

```

dsliceA = "dsliceA" + level1+ index1;
level3 = 4.5*level1;
v1 = 900/n;
v2 = v1*1/3;
v3 = v2*4/5;
_root[dsliceA]._alpha = 40;
_root[dsliceA]._width = v3;
_root[dsliceA]._x = 30+ v1*index1 + v3/2;
_root[dsliceA]._y = 460- level3;
level1++;

```

[6 프레임]

```

_root[dsliceA]._alpha = 100;
gotoAndPlay (4);

```

[7 프레임]

```

if (index1<n) {gotoAndPlay (3);
} else {gotoAndPlay (8);}

```

```

dtextA = "dtextA" + level1 + index1;
_root.attachMovie("text1", dtextA, depth++);
level3 = 4.5*level1;
v1 = 900/n;
v2 = v1*1/3;
v3 = v2*4/5;
_root[dtextA]._x = 30+ v1*index1 + v3/2 + 10;
_root[dtextA]._y = 440- level3;
_root[dtextA].field1 = value2;
index1++;

```

8, 9, 10, 11, 12 프레임에 group[1] 자료의 3d 애니메이션 바의 복제 및 위치 지정 코드, 수치 레이블의 복제 및 위치 지정 코드 등을 삽입한다. 다음은 8 프레임 - 12 프레임에 삽입한 스크립트 코드이다.

[8 프레임]

```

value3= data2A[index2];
value4 = data2[index2];
for (var i=0; i<value3;i++) {
    dsliceB= "dsliceB" + i + index2;
    _root.attachMovie ("slice2", dsliceB, depth++);
    _root[dsliceB]._x = -150;
    _root[dsliceB]._y = 150;
}
level2 =0;

```

[9 프레임]

```
if (level2<= value3) {gotoAndPlay (10);  
} else {gotoAndPlay (12);}
```

[10 프레임]

```
dsliceB = "dsliceB" + level2 + index2;  
level3 = 4.5*level2;  
v1 = 900/n;  
v2 = v1*1/3;  
v3 = v2*4/5;  
_root[dsliceB]._alpha = 40;  
_root[dsliceB]._width = v3;  
_root[dsliceB]._x = 30 + v1*index2+ v2+v3/2 ;  
_root[dsliceB]._y = 460- level3;  
level2++ ;
```

[11 프레임]

```
_root[dsliceB]._alpha = 100;  
gotoAndPlay (9);
```

[12 프레임]

```
if (index2<n) {gotoAndPlay (8);  
} else {gotoAndStop (13);}  
dtextB = "dtextB" + level2+ index2;  
_root.attachMovie("text1", dtextB, depth++ );  
level3 = 4.5*level2;  
v1 = 900/n;  
v2 = v1*1/3;
```



```

v3 = v2*4/5;
_root[dtextB]._x = 30 + v1*index2 + v2+ v3/2 + 10;
_root[dtextB]._y = 440 - level3;
_root[dtextB].field1 = value4;
index2++;

```

13 프레임에 차트의 제목 출력 코드, 그룹 레이블 출력 코드, 데이터의 레이블 복제 및 위치 지정 코드 등을 입력하면 BarChart3 무비의 제작이 완결된다. 아래의 스크립트 코드는 13 프레임에 삽입한 코드이다.

```

chartTitle = title;
group0 = group[0];
group1 = group[1];
for (var i = 0; i < n; i++) {
    v1 = 900/n;
    v2 = v1*1/4;
    dtextC= "dtextC" + i;
    _root.attachMovie ( "text2", dtextC, depth++ );
    _root[dtextC]._y = 480;
    _root[dtextC]._x = 30+ v1*i + v2*3/2;
    _root[dtextC].field2= label[i];
}

```

다음 데이터는 2003학년도 대학수학능력시험 결과 분석표이다(출처: 중앙교육진흥연구소). 이 데이터의 분석 결과를 BarChart3 무비로 표현하려면 Title 필드에 제목을 입력하고 group 필드에 재학생, 졸업생을 입력하고 Label 필드와 Data1, Data2 필드에 각각 계열과 영역별의 점수 콤마로 분

리하여 입력한 후에 drawBar 버튼을 누르면 된다

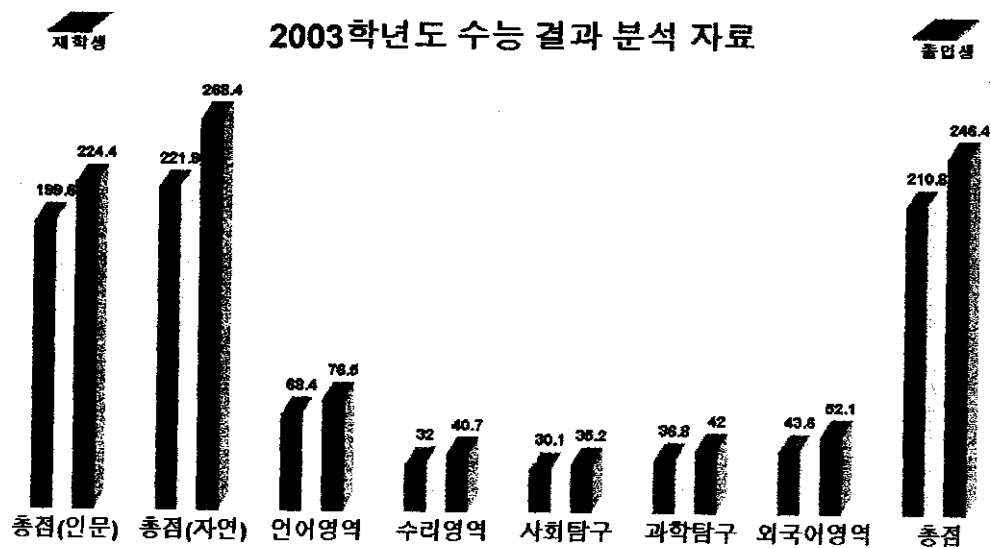
2003학년도 대학수학능력시험 결과 분석 자료표

(재학생, 졸업생 평균점수 비교)

총 점		재학생, 졸업생 총점 평균				언어 영역	
재학생	졸업생	인문계		자연계			
210.8	246.4	재학생	졸업생	재학생	졸업생	재학생	졸업생
		199.6	224.4	221.9	268.4	68.4	76.5
수리 영역		사회탐구 영역		과학탐구 영역		외국어 영역	
재학생	졸업생	재학생	졸업생	재학생	졸업생	재학생	졸업생
32.0	40.7	30.1	35.2	36.8	42.0	43.6	52.1

(인문계 자연계의 평균점수. 단, 예·체능계열은 분석자료에서 제외)

아래 그림은 2003학년도 대학수학능력시험 결과를 BarChart3 무비를 사용한 자료 분석 그래프이다.



재학생과 졸업생의 차이를 보면 전체집단의 경우 자연계는 46.5점, 인문계 24.8점으로 작년(2002년)에 비해 재학생과 졸업생의 평균에서 더 많은 차이를 보이고 있다. 전 계열에서 지난해에 비해 재학생과 졸업생의 점수차가 늘어난 것으로 나타났다.

또, 인문계에 비해 자연계열의 점수가 재학생과 졸업생의 점수차가 더 크다는 것을 눈으로 쉽게 알 수 있다.

참고 문헌

1. 부산대학교 수학교육과(2001), 웹 스크립팅과 수학교육.
2. 부산대학교 수학교육과(2003), 플래시 무비와 수학교육.
3. J. Allaire (2002), Macromedia Flash MX-A next-generation rich client, Macromedia, San Francisco, CA.
4. Macromedia (2000), Flash 5: Using Flash, Macromedia, San Francisco, CA.
5. Macromedia (2000), Flash 5: ActionScript Reference Guide, Macromedia, San Francisco, CA.
6. Macromedia (2002), Flash MX Tutorials, Macromedia, San Francisco, CA.
7. Netscape (1999), Core JavaScript Reference, Netscape, Mountain View, CA.
8. C. Perfetti and J.M. Spool (2002), Macromedia Flash: A new hope for web applications, User Interface Engineering Bradford, MA.
9. M. Williams (2002), ActionScript Coding Standards, Macromedia, San Francisco, CA.

Using Flash Movies to Construct Dynamic Charts

Kwak, Dong Ok

Major in Mathematics Education
Graduate school of Education
Pusan National University

Abstract

In this thesis, we study the method of constructing dynamic charts using Flash Movies. In section 2, we construct two pie charts, an interactive static pie chart and an interactive animation pie chart, using the graphic capacities of the Flash and ActionScript language built-in the Flash. In section 3, using the Flash Movies, we construct an interactive static bar chart and two interactive dynamic bar charts which can analyze the expenditure of the whole urban households in 2002 and the results of Korean Scholastic Aptitude Test (National Examination) of the year 2003.