



CHAP 8. 그래피(2)

1

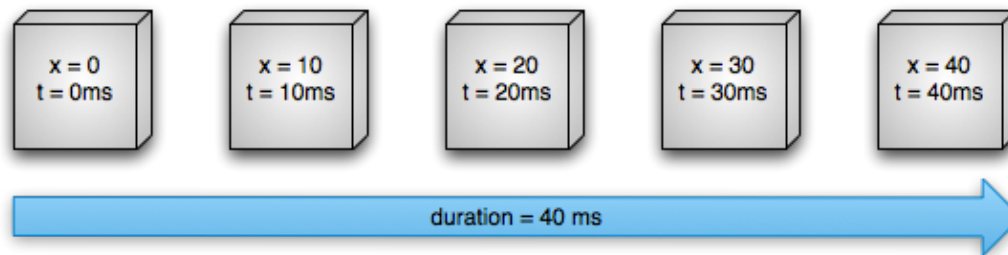
애니메이션

- 프로퍼티 애니메이션(property animation)
 - 모든 객체의 속성을 애니메이션할 수 있다.
- 뷰 애니메이션(view animation)
 - View 객체만을 애니메이션
- 드로워블 애니메이션(Drawable animation)
 - 여러 장의 이미지를 사용

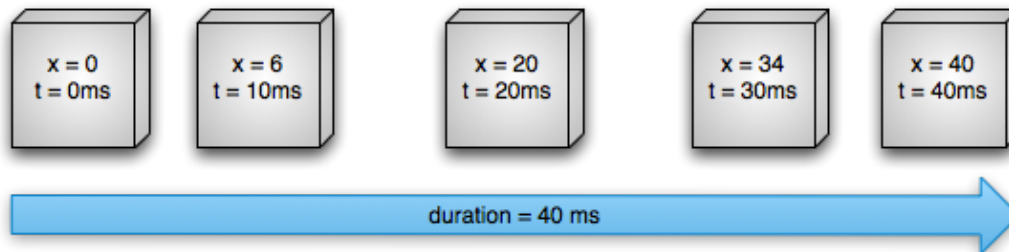
프로퍼티 애니메이션

- 다음과 같은 값들을 설정 가능
 - 지속 시간(duration)
 - 시간 보간(time interpolation): 시간의 함수로 값이 어떻게 변하는지를 정의
 - 반복 횟수와 반복 동작(Repeat count and behavior)
 - 애니메이터 집합(Animator sets): 애니메이션의 집합을 생성할 수 있다.
 - 프레임 재생 지연(Frame refresh delay): 애니메이션을 다시 그리는 횟수를 설정한다.

○ 선형 애니메이션 (AccelerateInterpolator)



○ 비선형 애니메이션 (AccelerateDecelerateInterpolator)



프러티 애니메이션 코드 (심심)

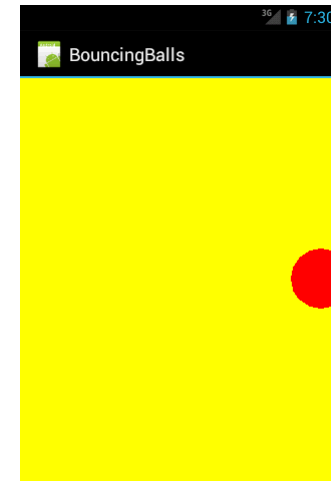
30에서 300까지 변화
되는 애니메이터 객체
를 생성한다.

애니메이션 값이 업데이트
될 때 호출된다. 여기서 현
재의 뷰를 다시 그리게 한
다.

애니메이션을 시작한
다.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_DOWN)
        return false;
    ValueAnimator moveAnim = ValueAnimator.ofFloat(30, 300);
    moveAnim.setDuration(2000);
    moveAnim.setInterpolator(new AccelerateInterpolator());
    moveAnim.addUpdateListener(new ObjectAnimator.AnimatorUpdateListener() {
        public void onAnimationUpdate(ValueAnimator animation) {
            mX = (Float) animation.getAnimatedValue();
            invalidate();
        }
    });
    moveAnim.start();
    return true;
}
```

```
@Override
protected void onDraw(Canvas canvas) {
    Paint paint = new Paint();
    paint.setColor(Color.RED);
    canvas.drawCircle(mX, 200, 30, paint);
}
```



```

class MyView extends View {
    float mX = 30;

    public MyView(Context context) {
        super(context);
        setBackgroundColor(Color.YELLOW);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() != MotionEvent.ACTION_DOWN)
            return false;
        ValueAnimator moveAnim = ValueAnimator.ofFloat(30, 480);
        moveAnim.setDuration(2000);
        moveAnim.setInterpolator(new AccelerateDecelerateInterpolator());
        moveAnim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
            public void onAnimationUpdate(ValueAnimator valueAnimator) {
                mX = (Float) valueAnimator.getAnimatedValue();
                invalidate();
            }
        });
        moveAnim.start();
        return true;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Paint paint = new Paint();
        paint.setColor(Color.RED);
        canvas.drawCircle(mX, cy: 200, radius: 30, paint);
    }
}

public class BouncingBallsActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(context: this));
    }
}

```

뷰 애니메이션

rotate.xml

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<set xmlns:android="http://schemas.android.com/apk/res/android"  
    android:shareInterpolator="false">
```

```
    <rotate
```

```
        android:interpolator="@android:anim/accelerate_decelerate_interpolator"  
        android:fromDegrees="0"  
        android:toDegrees="360"  
        android:pivotX="25%"  
        android:pivotY="25%"  
        android:duration="6000"
```

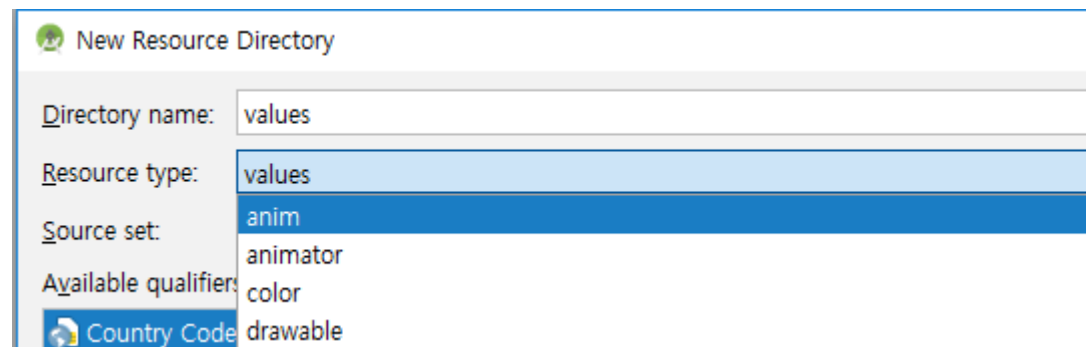
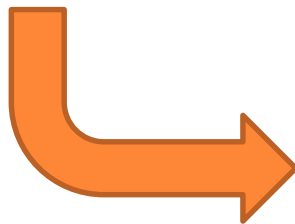
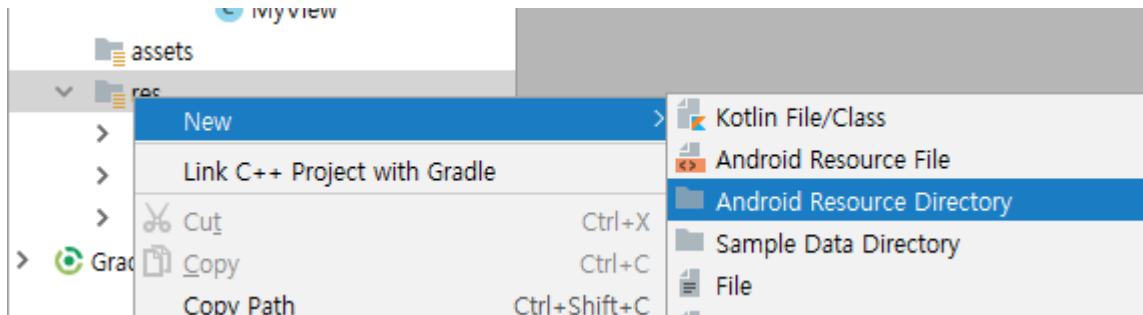
```
    />
```

```
</set>
```

360도 회전을 지시

- res/anim 디렉토리에 생성함

○ res/anim 디렉토리에 생성함



코드 (실습)

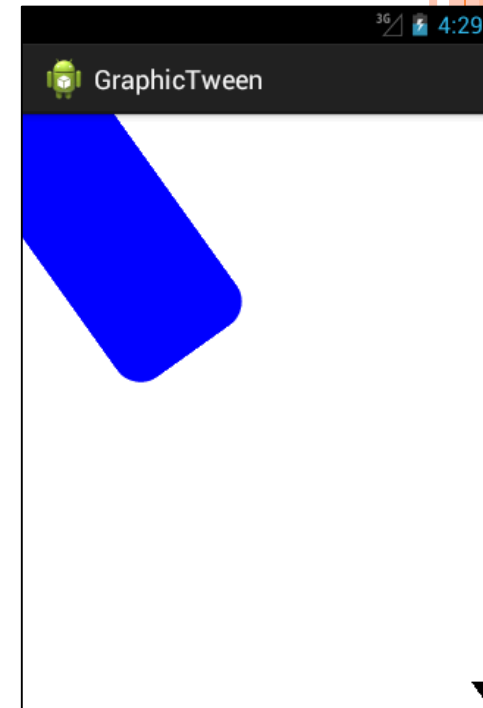
```
public class GraphicTweenActivity extends Activity {
    LinearLayout mLinearLayout;
    Animation anim;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 선형 레이아웃을 생성한다.
        mLinearLayout = new LinearLayout(this);

        float[] array = new float[] { 20, 20, 20, 20, 20, 20, 20, 20 };
        ShapeDrawable rect = new ShapeDrawable(new RoundRectShape(array, null,
            null));
        rect.setIntrinsicHeight(300);
        rect.setIntrinsicWidth(600);
        rect.getPaint().setColor(Color.BLUE);

        ImageView i = new ImageView(this);
        i.setImageDrawable(rect);
        i.setVisibility(View.VISIBLE);
        anim = AnimationUtils.loadAnimation(this, R.anim.rotate);
        i.startAnimation(anim);
        // ImageView를 레이아웃에 추가한다
        mLinearLayout.addView(i);
        setContentView(mLinearLayout);
    }
}
```



드로워블 애니메이션

- 영화 필름처럼 여러 개의 이미지가 순서대로 재생되어서 생성되는 전통적인 애니메이션



드로워블 애니메이션(실습)

- 애니메이션을 구성하는 프레임들을 나열하는 XML 파일을 생성(res/drawable 디렉토리에 생성)


```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true" >

    <item
        android:drawable="@drawable/rocket1"
        android:duration="200"/>

    <item
        android:drawable="@drawable/rocket2"
        android:duration="200"/>

    <item
        android:drawable="@drawable/rocket3"
        android:duration="200"/>

</animation-list>
```



```
public class DrawableAnimationActivity extends Activity {
    AnimationDrawable rocketAnimation;
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
        rocketImage.setBackgroundResource(R.drawable.rocket);
        rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
    }
```

애니메이션 리소스를
이미지 뷰의 배경으
로 설정한다.

애니메이션 객체
를 얻는다.

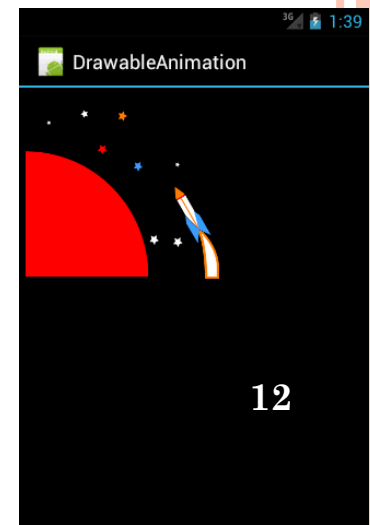
```
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            rocketAnimation.start();
            return true;
        }
        return super.onTouchEvent(event);
    }
}
```

화면이 터치되면
애니메이션을 시
작한다.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

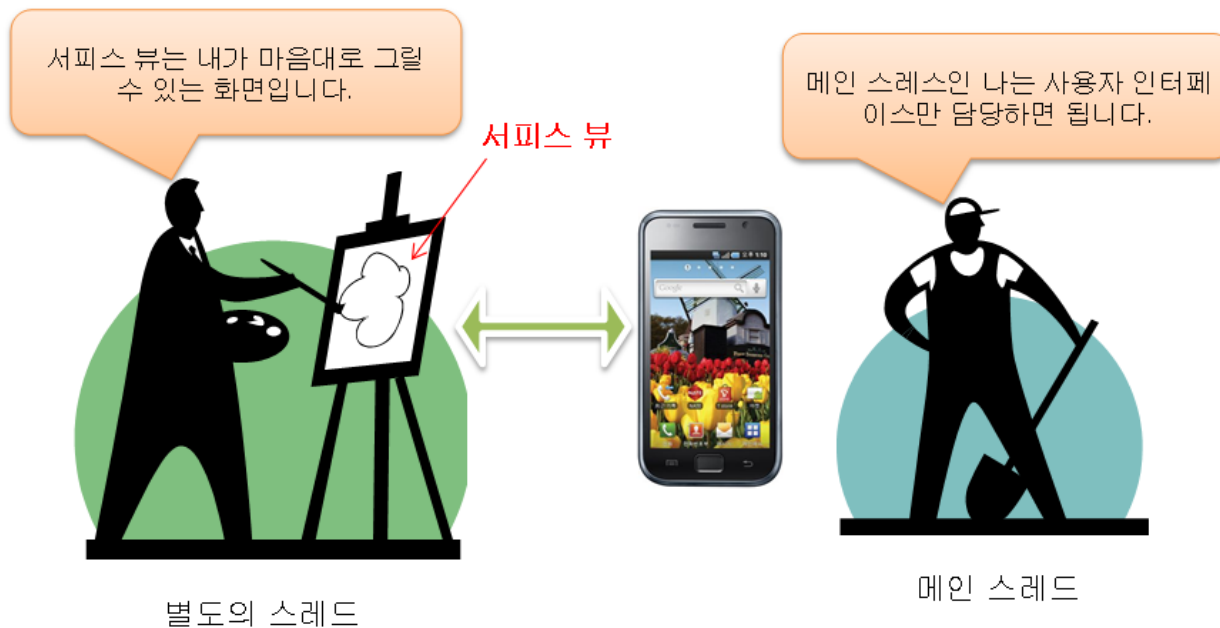
    <ImageView
        android:id="@+id/rocket_image"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```



서피스 뷰

- 서피스뷰는 사용자 인터페이스와는 별도로 애플리케이션에게 그림을 그릴 수 있는 화면을 제공



서피스 뷰의 구조

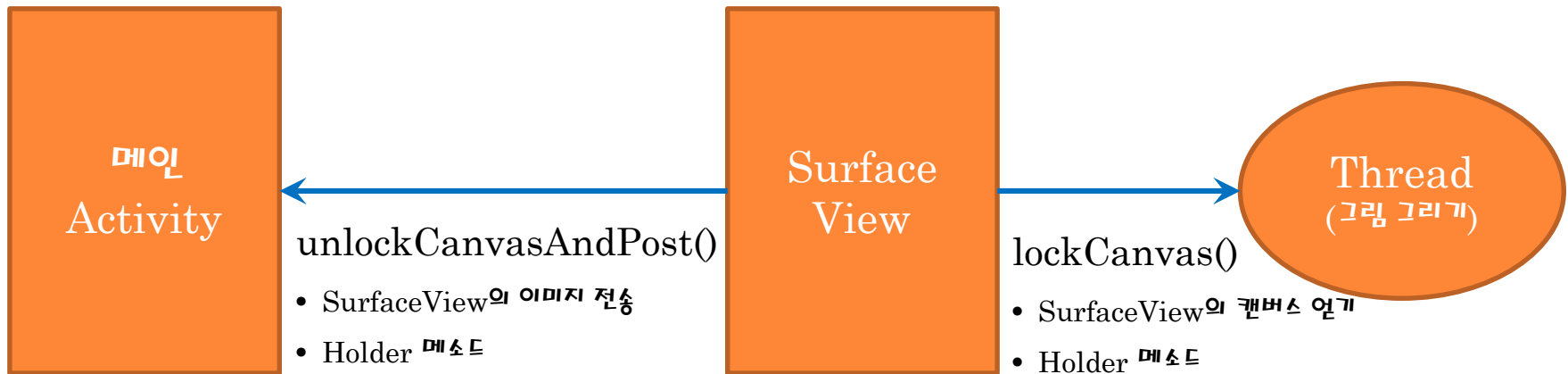
- SurfaceView 클래스를 상속한 뷰를 생성한다.

```
class MyView extends SurfaceView implements SurfaceHolder.Callback {  
    public void surfaceCreated(SurfaceHolder holder) {  
        // 서피스가 준비되었으므로 스레드를 시작한다.  
        ...  
    }  
    public void surfaceDestroyed(SurfaceHolder holder) {  
        // 서피스가 소멸되었으므로 스레드를 종료한다.  
        ...  
    }  
    public void surfaceChanged(SurfaceHolder holder, int format,  
                               int width, int height) {  
        // 서피스가 변경  
        ...  
    }  
}
```

서피스 뷰의 개념

getHolder()

- SurfaceView의 핸들러를 반환



surfaceCreated()

- SurfaceView의 캔버스가 준비되면 호출됨

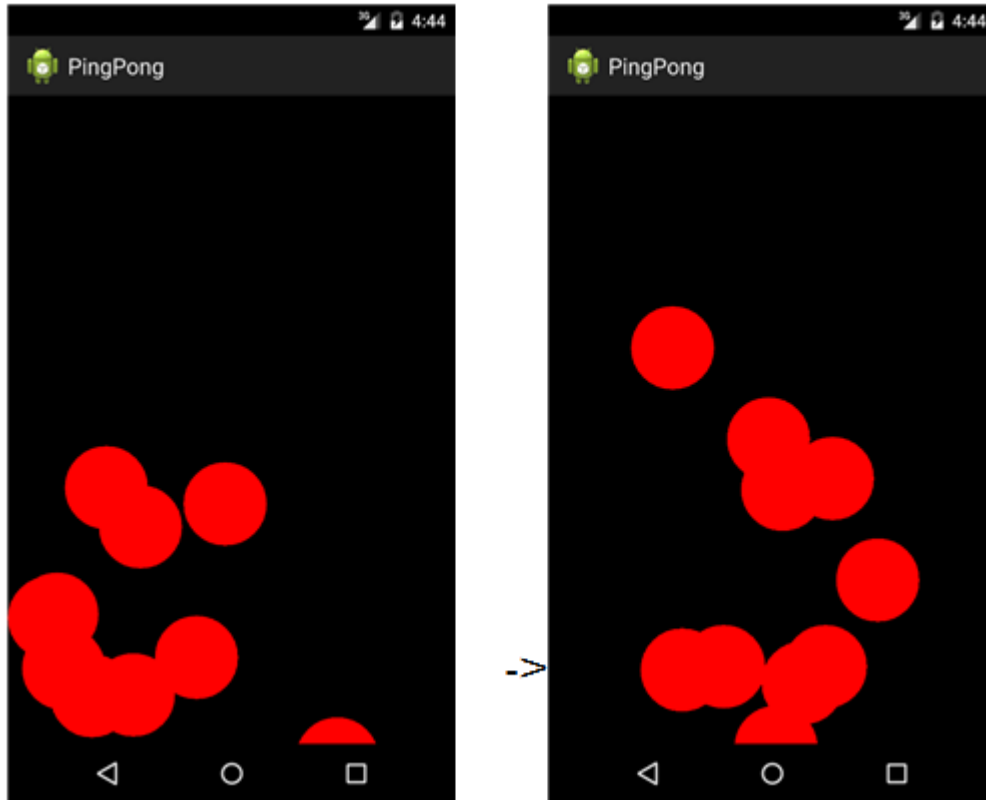
surfaceDestroyed()

surfaceChanged()

스레드를 정의한다.

```
class MyThread extends Thread {  
    SurfaceHolder holder;  
  
    ...  
    public void run()  
    {  
        canvas = holder.lockCanvas(); // 서비스의 캔버스를 반환  
        // 캔버스에 그림을 그린다.  
  
        ...  
        holder.unlockCanvasAndPost(canvas); // 서피스 캔버스의 그림을  
                                              // 화면에 그리기  
    }  
}
```


서피스 뷰 예제



핵심적인 코드

```
// 서피스 뷰 정의
public class MySurfaceView extends SurfaceView implements
    SurfaceHolder.Callback {
    ...
    public MySurfaceView(Context context) { // 생성자
        super(context);
        SurfaceHolder holder = getHolder(); // 서피스 뷰의 홀더를 얻는다.
        holder.addCallback(this); // 콜백 메소드를 처리한다.
        thread = new MyThread(holder); // 스레드를 생성한다.
        // Ball 객체를 생성하여서 배열에 넣는다.
        for (int i = 0; i < 10; i++)
            basket[i] = new Ball(20);
    }
    public MyThread getThread() {
        return thread;
    }
}
```

핵심적인 코드

```
public void surfaceCreated(SurfaceHolder holder) {  
    // 스레드를 시작한다.  
    thread.setRunning(true);  
    thread.start();  
}  
  
public void surfaceDestroyed(SurfaceHolder holder) {  
    boolean retry = true;  
    // 스레드를 중지시킨다.  
    thread.setRunning(false);  
    while (retry) {  
        try {  
            thread.join(); // 메인 스레드와 합친다.  
            retry = false;  
        } catch (InterruptedException e) { }  
    }  
}  
  
public void surfaceChanged(SurfaceHolder holder, int format,  
    int width,    int height) {  
}
```

핵심적인 코드

```
public class MyThread extends Thread {  
    ...  
    @Override  
    public void run() {  
        while (mRun) {  
            Canvas c = null;  
            try {  
                c = mSurfaceHolder.lockCanvas(null);  
                c.drawColor(Color.BLACK);    // 캔버스의 배경을 지운다.  
                synchronized (mSurfaceHolder) {  
                    for (Ball b : basket) { // basket의 모든 원소를 그린다.  
                        b.paint(c);  
                    }  
                }  
            } finally {  
                if (c != null) {    mSurfaceHolder.unlockCanvasAndPost(c);    }  
            }  
        }  
    }  
}
```

```

public class PingPongActivity extends Activity {
    MySurfaceView view;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Display display = getWindowManager().getDefaultDisplay();
        Point size = new Point();
        display.getSize(size);
        Ball.WIDTH = size.x;
        Ball.HEIGHT = size.y;

        view = new MySurfaceView(context, this);
        setContentView(view);
    }
}

```

```

class Ball {
    int x, y, xInc = 1, yInc = 1;
    int radius;
    static int WIDTH = 1080, HEIGHT = 1920;

    public Ball() {
        radius = WIDTH/20;

        x = (int) (Math.random() * (WIDTH - radius));
        y = (int) (Math.random() * (HEIGHT - radius));
        x = x < radius ? radius : x;
        y = y < radius ? radius : y;

        xInc = (int) (Math.random() * 30 + 1);
        yInc = (int) (Math.random() * 30 + 1);
    }

    public void paint(Canvas g) {
        Paint paint = new Paint();

        if (x < radius || x > (WIDTH - radius))
            xInc = -xInc;
        if (y < radius || y > (HEIGHT - radius))
            yInc = -yInc;

        x += xInc;
        y += yInc;

        paint.setColor(Color.RED);
        g.drawCircle(x, y, radius, paint);
    }
}

```

```

public class MySurfaceView extends SurfaceView implements SurfaceHolder.Callback {
    public Ball basket[] = new Ball[10];
    private MyThread thread;

    public MySurfaceView(Context context) {
        super(context);

        SurfaceHolder holder = getHolder();
        holder.addCallback(this);

        thread = new MyThread(holder);

        for (int i = 0; i < 10; i++)
            basket[i] = new Ball();
    }

    public void surfaceCreated(SurfaceHolder holder) {
        thread.setRunning(true);
        thread.start();
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        boolean retry = true;

        thread.setRunning(false);
        while (retry) {
            try {
                thread.join();
                retry = false;
            } catch (InterruptedException e) {
            }
        }
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    }
}

```

```

public class MyThread extends Thread {
    private boolean mRun = false;
    private SurfaceHolder mSurfaceHolder;

    public MyThread(SurfaceHolder surfaceHolder) {
        mSurfaceHolder = surfaceHolder;
    }

    @Override
    public void run() {
        while (mRun) {
            Canvas c = null;
            try {
                c = mSurfaceHolder.lockCanvas(null);
                c.drawColor(Color.BLACK);
                synchronized (mSurfaceHolder) {
                    for (Ball b : basket) {
                        b.paint(c);
                    }
                }
            } finally {
                if (c != null) {
                    mSurfaceHolder.unlockCanvasAndPost(c);
                }
            }
        }
    }

    public void setRunning(boolean b) {
        mRun = b;
    }
}

```

서피스 뷰 예제 (실습)

