Ball

2014-12-21

이승진

**목차**

[1. Ball 프로젝트 생성 2](#_Toc406975852)

[2. Timer1Activity 생성 3](#_Toc406975853)

[1) activity\_timer1.xml 수정 4](#_Toc406975854)

[2) Timer1Activity.java 수정 6](#_Toc406975855)

[3. MainActivity 수정 9](#_Toc406975856)

[1) activity\_main.xml 수정 9](#_Toc406975857)

[2) MainActivity.java 수정 9](#_Toc406975858)

[4. 실행 10](#_Toc406975859)

[5. Play1Activity 생성 11](#_Toc406975860)

[1) Play1Activity.java 수정 11](#_Toc406975861)

[6. MainActivity 수정 16](#_Toc406975862)

[1) activity\_main.xml 수정 16](#_Toc406975863)

[2) MainActivity.java 수정 16](#_Toc406975864)

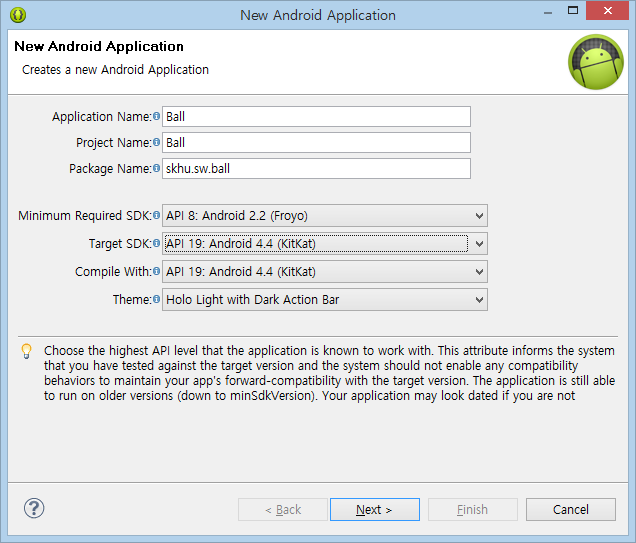
[7. 실행 17](#_Toc406975865)

[8. Play2Activity 생성 18](#_Toc406975866)

[1) Play2Activity.java 수정 18](#_Toc406975867)

예제 소스는 강의노트 웹폴더의 ball.zip 파일

# Ball 프로젝트 생성



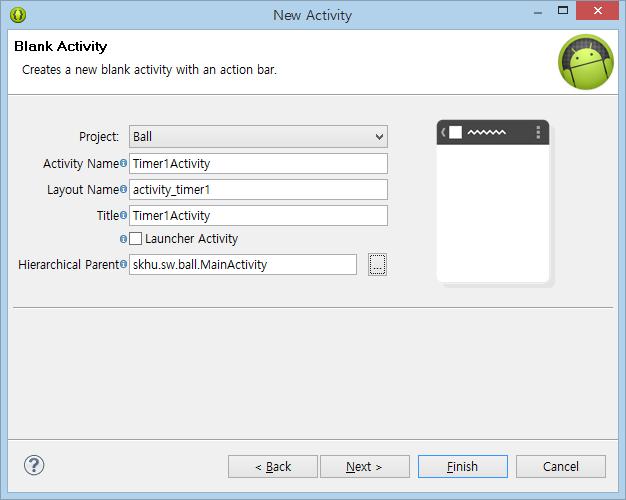
Project Name: Ball

Package Name: skhu.sw.ball

# Timer1Activity 생성

게임 구현에 필요한 타이머 기능을 먼저 구현해 보자.

메뉴: File – New – Other – Android – Android Activity



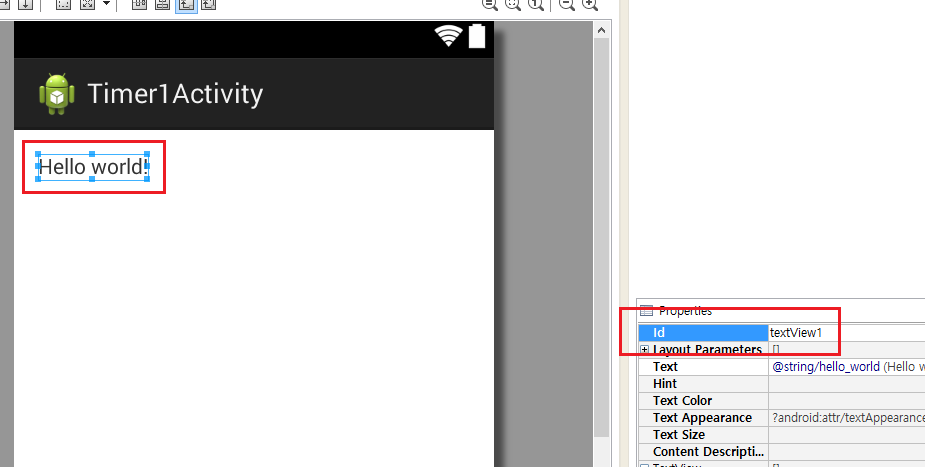
Activity Name: Timer1Activity

Hierarchical Parent: skhu.sw.ball.MainActivity

## activity\_timer1.xml 수정

Timer1Activity의 레이아웃 리소스 XML 파일인

res/layout/activity\_timer1.xml을 수정하자

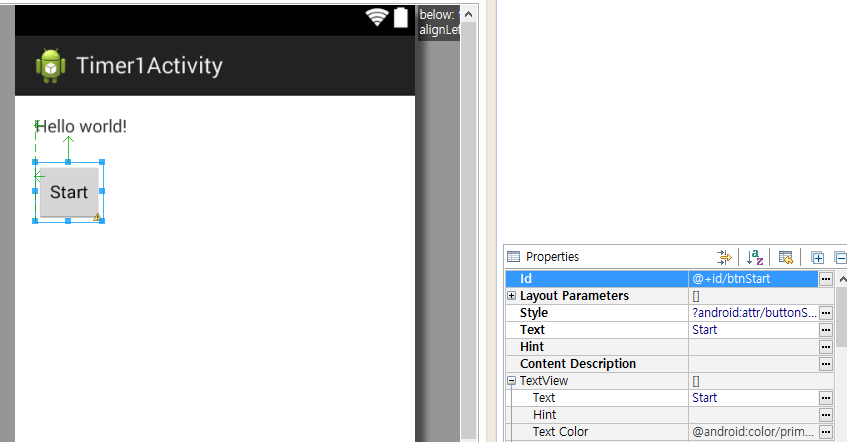


위와같이 TextView의 id 속성값에 textView1 을 입력하자.

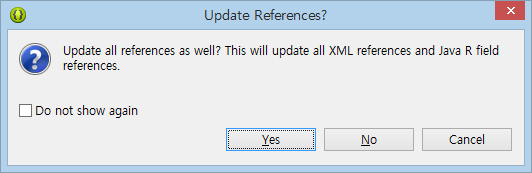
버튼을 하나 추가하고, 다음과 같이 속성을 입력하자

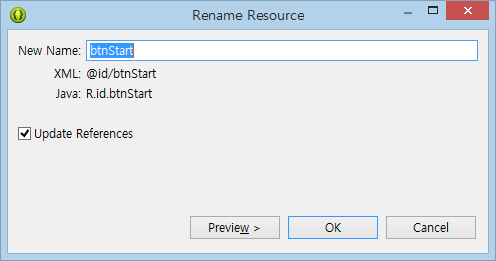
id : btnStart

Text : Start



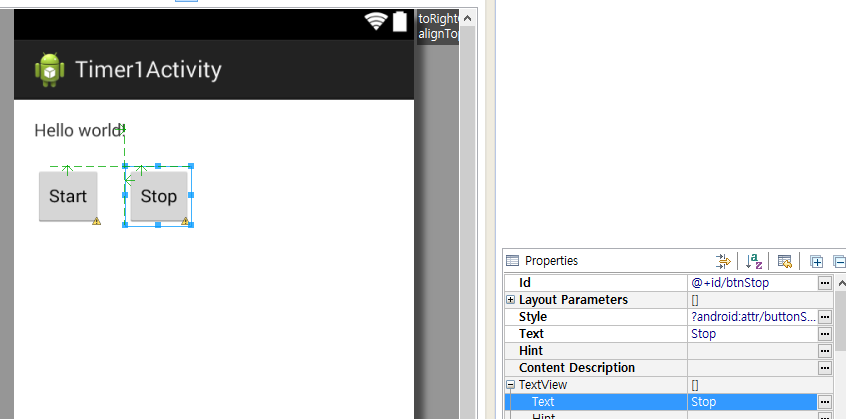
위와 같이 id 속성값을 변경할 때, "Update References?" 라고 묻는 대화상자가 나타나면 "Yes"를 누르자.





여기에서도 OK 버튼을 누르자.

뷰(View) 객체의 id를 변경하면 XML 파일의 @id/textView1 부분과 Java 소스코드의 R.id.textView1 부분이 둘 다 변경되어야 한다. 이 변경(update)을 자동으로 해주겠다고 묻는 대화상자이다.



위와 같이 버튼을 하나 더 추가하고 다음과 같이 속성을 입력하자.

id : btnStop

Text : Stop

## Timer1Activity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | package skhu.sw.ball;  import android.support.v7.app.ActionBarActivity;  import android.os.Bundle;  import android.os.Handler;  import android.os.Message;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  import android.widget.TextView;  public class Timer1Activity extends ActionBarActivity {  static final int TIMER\_INTERVAL = 1000;  Handler handler;  TextView textView1;  int count;    @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_timer1);  textView1 = (TextView)findViewById(R.id.textView1);    Button btnStart = (Button)findViewById(R.id.btnStart);  btnStart.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  startTimer();  }  });    Button btnStop = (Button)findViewById(R.id.btnStop);  btnStop.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View arg0) {  stopTimer();  }  });  }    void startTimer() {  handler = new Handler() {  public void handleMessage(Message msg) {  ++count;  textView1.setText("count# " + count);  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);//0은 의미없음.  }  };  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);  }    void stopTimer() {  handler.removeCallbacksAndMessages(null);//메시지전달 취소  handler = null;//자바는 free가 없다. null을 넣어놓아야 사라짐 참조되지 않는객체는 사라짐  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.timer1, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄16) 메소드가 자동으로 호출될 시간인 1000 밀리초(mili second)를 상수(constant)로 정의하였다.

(줄17) 정해진 시간 간격으로 메소드를 자동으로 호출하기 위해 Handler 객체를 사용한다.

(줄18) 정해진 시간 각격으로 호출된 메소드에서 이 멤버 변수가 가르키는 TextView에 카운트 값을 표시할 것이다. 줄25에서 이 멤버 변수에 TextView 객체에 대한 참조를 대입한다.

(줄19) TextView에 표시할 카운트 값을 저장할 멤버 변수이다. 줄47~48에서 이 카운트 값이 TextView에 채워진다.

(줄27~33) Start 버튼에 OnClickListener를 등록하였다. Start 버튼이 클릭되면, 줄30의 onClick 메소드가 호출되어, 줄31에서 startTimer 메소드가 호출될 것이다.

(줄35~41) Stop 버튼에 OnClickListener를 등록하였다. Stop 버튼이 클릭되면, 줄38의 onClick 메소드가 호출되어, 줄39에서 stopTimer 메소드가 호출될 것이다.

(줄44) Start 버튼이 클릭되면 이 startTimer 메소드가 호출된다.

정해진 시간 간격으로 메소드를 자동으로 호출해주는 기능(타이머 기능)을 시작하는 메소드이다.

Handler 객체를 이용하여 이 기능을 구현한다.

(줄45~51) anonymous inner class 객체를 생성하는 코드이다.

Handler 클래스의 객체가 하나 생성되어 멤버 변수 handler에 대입되고 있다.

(줄46~50) 줄45에서 생성된 Handler 객체의 메소드가 정의되었다.

메소드가 정의되었을뿐이고, 아직 이 메소드는 호출되지 않는다.

누군가 Handler 객체에 메시지(message)를 전달하면 이 메소드가 호출된다.

아직 아무도 메시지를 전달하지 않았기 때문에 아직 이 메소드는 호출되지 않는다.

(줄52) Handler 클래스의 sendEmptyMessagDelayed 메소드 호출이다.

줄45에서 생성된 핸들러 객체에 메시지를 하나 전달하는 메소드 호출이다.

그런데 메시지가 즉시 전달되는 것이 아니고, 둘째 파라미터로 주어진 밀리초만큼 지체하다가 전달된다.

그래서 1초 후에 핸들러 객체에 메시지가 하나 전달된다.

핸들러 객체에 메시지가 전달되면 줄46의 handleMessage 메소드가 저절로 호출된다.

즉 1초 후에 줄46의 handleMessage 메소드가 저절로 호출된다.

(줄47~49) 1초 후에 handleMessage 메소드가 저절로 호출되어 이 코드가 실행된다.

(줄47~48) count 변수 값이 1증가되어 TextView에 표시된다.

(줄49) 줄45에서 생성된 핸들러 객체에 메시지를 하나 전달하는 메소드 호출이다.

그런데 메시지가 즉시 전달되는 것이 아니고 1초 후에 전달된다.

1초 후에 줄46의 handleMessage 메소드가 또 저절로 호출되게 된다.

이 일련의 과정이 1초의 시간 간격으로 계속 반복된다.

1초 간격으로 텍스트뷰에 카운트 값이 1씩 증가하며 표시된다.

(줄55) 사용자가 Stop 버튼을 클릭하면 줄55의 stopTimer 메소드가 호출된다.

(줄56) 줄45에서 생성된 핸들러 객체를 멤버 변수 handler가 참조하고 있다. 이 핸들러 객체에 전달되기 위해 대기중인 메시지들을 전부 취소한다.

(줄57) 멤버 변수 handler에 null을 대입하면, 이제 멤버 변수 handler는 핸들러 객체를 참조하지 않게 된다.

줄45에서 생성된 핸들러 객체는 더 이상 참조되고 있지 않으므로, 메모리에서 제거된다.

핸들러 객체가 제거되었으므로 줄46의 handleMessage 메소드는 더 이상 호출되지 않는다.

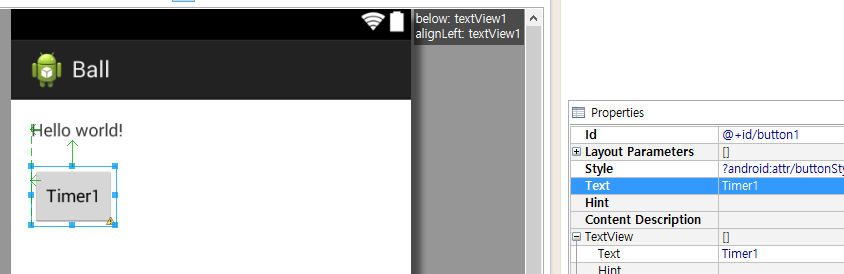
즉 타이머 기능이 정지된다.

# MainActivity 수정

게임 화면인 Play1Activity로 넘어가기 위한 버튼을 MainActivity에 추가하자.

## activity\_main.xml 수정

MainActivity의 화면 레이아웃인 res/layout/activity\_main.xml 파일을 열어서 버튼을 하나 추가하자.



Timer1Activity를 호출하기 위한 버튼을 하나 추가하자.

## MainActivity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | package skhu.sw.ball;  import android.support.v7.app.ActionBarActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  public class MainActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  Button button1 = (Button)findViewById(R.id.button1);  button1.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View v) {  Intent intent = new Intent(MainActivity.this, Timer1Activity.class);  startActivity(intent);  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄19~26) 버튼에 온클릭리스너 객체를 하나 등록한다.

이 버튼이 클릭되면 줄22의 onClick 메소드가 호출된다.

(줄23~24) Timer1Activity 액티비티가 호출된다.

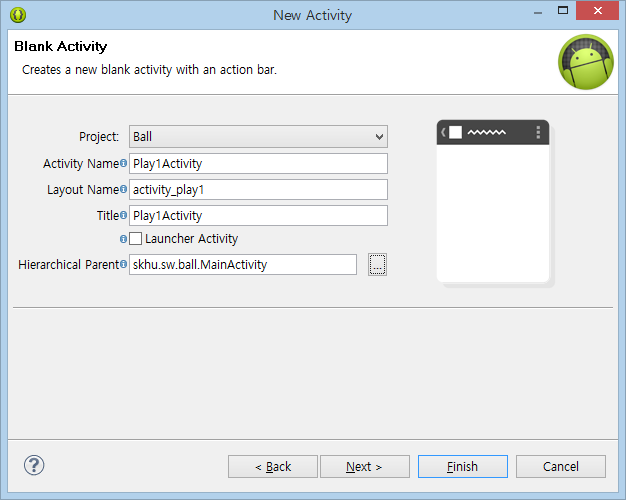
# 실행

Timer1Activity를 실행하여 타이머 기능을 확인해 보자.

# Play1Activity 생성

게임 화면을 구현하자.

메뉴: File – New – Other – Android - Android Activity



Activity Name: Play1Activity

Hierarchical Parent: skhu.sw.ball.MainActivity

## Play1Activity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124 | package skhu.sw.ball;  import android.support.v7.app.ActionBarActivity;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.graphics.Paint;  import android.graphics.RadialGradient;  import android.graphics.Shader.TileMode;  import android.os.Bundle;  import android.os.Handler;  import android.os.Message;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  public class Play1Activity extends ActionBarActivity {    MyView1 myView;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  myView = new MyView1(this);  setContentView(myView);  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.play1, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }      @Override  protected void onResume() {  super.onResume();  myView.startPlay();  }  @Override  protected void onPause() {  super.onPause();  myView.stopPlay();  }    }  class MyView1 extends View {    static final int TIMER\_INTERVAL = 15;  Handler handler;  Ball ball;    public MyView1(Context context) {  super(context);  ball = new Ball();  }    public void startPlay() {  handler = new Handler() {  public void handleMessage(Message msg) {  ball.move(getWidth(), getHeight());  invalidate();  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);  }  };  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);  }    public void stopPlay() {  handler.removeCallbacksAndMessages(null);  handler = null;  }    public void onDraw(Canvas canvas) {  ball.draw(canvas);  }    class Ball {  static final int RADIUS = 20;  int x, y;  int vx = 8, vy = 8;    public void move(int screenWidth, int screenHeight) {  x = x + vx;  y = y + vy;    if (x < 0) {  x = 0 - x;  vx = Math.abs(vx);  } else if (ball.x >= screenWidth) {  x = screenWidth - (x - screenWidth);  vx = -Math.abs(vx);  }  if (y < 0) {  y = 0 - y;  vy = Math.abs(vy);  } else if (ball.y >= screenHeight) {  y = screenHeight - (y - screenHeight);  vy = -Math.abs(vy);  }  }    void draw(Canvas canvas) {  Paint paint = new Paint();  RadialGradient gradient = new RadialGradient((x - RADIUS / 3), (y - RADIUS / 3), RADIUS, Color.WHITE, Color.BLUE, TileMode.CLAMP);  paint.setShader(gradient);  canvas.drawCircle(x, y, RADIUS, paint);  }    }    } |

(줄59~124) MyView1 클래스를 구현하였다.

지금까지 사용했던 Button, TextView, EditView 클래스의 부모 클래스가 View 클래스이다.

이 View 클래스를 상속하여 MyView1 클래스를 구현하였다.

게임 화면을 구현한 뷰 클래스이다.

(줄24) MyView1 클래스의 객체를 생성하여 액티비티의 멤버 변수에 대입한다.

(줄25) 생성된 MyView1 클래스 객체를 액티비티의 내용으로 설정한다.

지금까지 액티비티의 내용에 해당하는 뷰 객체는 레이아웃 XML 리소스 파일로부터 자동생성하였다.

줄24~25처럼 직접 뷰 객체를 자바 코드에서 직접 생성(new)하여 액티비티에 채울 수도 있다.

(줄47) 액티비티가 화면에 보이기 시작할 때 액티비티의 onResume 메소드가 호출된다.

이때부터 게임의 애니메이션이 시작되도록 MyView 클래스의 startPlay() 메소드를 호출한다.

(줄48) 액티비티가 화면 아래로 내려가면 액티비티의 onPause 메소드가 호출된다.

이때 게임의 애니메이션은 중지되도록 MyView 클래스의 stopPlay() 메소드를 호출한다.

(줄58) View 클래스를 상속하여 MyView1 클래스를 구현하였다.

(줄60) 애니메이션 장면의 시간 간격 밀리초(mili second)를 상수(constant)로 정의하였다.

15밀리초 간격으로 애니메이션 장면들이 진행된다.

(줄61) 정해진 시간 간격으로 메소드를 자동으로 호출하기 위해 Handler 객체를 사용한다.

게임의 애니메이션을 구현한 메소드가 15밀리초 간격으로 반복해서 자동 호출되어야 한다.

이것을 Handler 객체를 사용해서 구현하려고 한다.

(줄62) 화면에서 움직일 볼 객체이다.

(줄64) View 클래스를 상속 받아 자식 클래스를 구현할 때는 반드시 생성자 파라미터가 Context 클래스인

이 생성자를 구현해야 한다. Context 클래스는 Activity 클래스의 부모 클래스이다. 줄24에서 MyView1 객체를 생성할 때 이 생성자를 호출하고 있다. 생성자 파라미터로 액티비티 객체를 전달하고 있다.

(줄65) View 클래스를 상속 받아 자식 클래스를 구현할 때, 반드시 보무 클래스의 생성자를 이렇게 호출하여야 한다.

(줄66) 화면에서 움직일 볼 객체를 생성하여 멤버 변수에 대입한다.

(줄69) startPlay는 애니메이션을 시작하기 위한 메소이다.

이 메소드는 액티비티 클래스의 onResume 메소드에서 호출된다 (줄47)

(줄71) 애니메이션 시간 각격마다 반복해서 자동으로 메소드를 호출해 주는 Handler 객체를 생성하여 멤버 변수에 대입한다.

(줄70~76) anonymous inner class 문법으로 Handler 클래스의 자식 클래스의 객체를 하나 생성하여 멤버 변수에 대입한다.

(줄71~75) 줄70에서 생성된 핸들러 객체의 메소드이다. 이 handleMessage 메소드는 정해진 시간 간격으로 반복해서 호출된다.

(줄72) 볼 객체의 move 메소드를 호출하여 볼의 좌표를 조금 이동한다.

getWidth 메소드는 View 클래스의 메소드이다. View의 폭(width)를 리턴한다.

getHeight 메소드는 View의 높이(height)를 리턴한다.

(줄73) 뷰(view)의 화면을 새로 그리라는 명령에 해당하는 메소드이다.

invalidate 메소드를 호출하면 화면에서 뷰가 차지하고 있는 영역이 지워지고, 뷰 객체의 onDraw 메소드가 호출된다. onDraw 메소드에서 뷰의 화면을 전부 그려야 한다.

(줄85) 뷰를 화면에 그리기 위한 메소드이다. 이 메소드는 보통 직접 호출되지 않고, invalidate 메소드 호출을 통해서 간접 호출된다. 이 메소드에 뷰의 화면을 그리는 코드를 구현해야 한다.

(줄86) 화면에 볼 객체 하나를 그린다.

(줄89~121) 화면에서 움직일 볼을 구현한 클래스이다.

Play1 클래스 내부에서만 Ball 클래스를 사용할 것이라서 Play1 클래스의 내부 클래스(inner class)로 구현하였다.

(줄90) 볼의 반지름을 상수(constant)로 정의하였다.

(줄91) 볼의 현재 x, y 좌표에 해당하는 멤버 변수

(줄92) 볼의 이동 속도에 해당하는 멤버 변수

(줄94) 화면에서 볼이 움직이는 좌표 계산을 구현하였다.

이 메소드가 호출될 때 마다 볼이 볼의 좌표(x, y)가 조금씩 이동한다.

볼이 움직일 영역 화면 크기(width, height)가 파라미터로 주어진다.

볼의 x 좌표는 0보다 커야하고 화면의 width보다 작아야 한다.

볼의 y 좌표는 0보다 커야하고 화면의 height보다 작아야 한다.

화면 왼쪽 상단의 좌표가 (0, 0) 이다.

x좌표는 오른쪽으로 증가하고 y좌표는 아래쪽으로 증가한다.

(줄95,96) 볼의 x, y 좌표가 vx, vy 멤버 변수 값만큼 이동한다.

(줄98) 볼의 x좌표가 화면의 왼쪽 경계 밖으로 나갔으면

(줄99) 볼이 왼쪽벽에 튕겨 나와야 하므로, 왼쪽으로 나간 만큼 화면 오른쪽으로 이동한다.

x 값이 음수이므로 (0 – x) 은 양수이다.

(줄100) vx 값을 양수로 바꿔준다. vx가 양수이면 볼의 x좌표는 점점 증가하게 되어 볼이 오른쪽으로 이동한다.

(줄101) 볼의 x좌표가 화면의 오른쪽 경계 밖으로 나갔으면

(줄102) 볼이 오른쪽벽에 튕겨 나와야 하므로, 오른쪽으로 나간 만큼 화면 왼쪽으로 이동한다.

볼이 오른쪽으로 나간 만큼은 (x – screenWidth) 이다.

그만큼 왼쪽으로 이동하면 screenWidth – (x – screenWidth) 이다.

(줄103) vx 값을 음수로 바꿔준다. vx가 음수이면 볼의 x좌표는 점점 감소하게 되어 볼이 왼쪽으로 이동한다.

(줄114) 볼을 화면에 그리는 메소드

(줄115) 색을 어떻게 칠하고 선을 어떻게 그릴지 설정할 때 Paint 객체를 생성한다.

(줄116) 어느 한 점에서 시작해서 점점 멀어지면서 색이 변하는 방식으로 색을 칠하는 방식을 지정할 때,

RadialGradient 객체를 생성한다.

RadialGradient 생성자의 첫째 파라미터= 시작점 x좌표, 둘째 파라미터= 시작점 y좌표

셋째 파라미터= 반지름, 넷째 파라미터= 시작점에서의 색,

다섯째 파라미터= 시작점에서 반지름만큼 떨어진 지점의 색.

여섯째 파라미터= 시작점에서 반지름보더 더 먼 지점을 어떻게 칠할지 지정한다.

여섯째 파라미터 값이 TileMode.CLAMP이면 다섯째 파라미터 색으로 계속 칠해진다.

(줄116) 볼의 중심점에서 약간 왼쪽 위인 것을 시작점으로 흰색에서 파란색으로 점점 변하며 칠하기 위해 RadialGradient 객체를 생성하였다.

(줄118) 칠하는 방식을 설정한 RadialGradient 객체를 paint 객체에 설정하였다.

(줄119) 캔버스(canvas) 객체의 drawCircle 메소드를 호출하여 원을 그린다.

첫째 파라미터= 원의 x 좌표

둘째 파라미터= 원의 y 좌표

셋째 파라미터= 원의 반지름

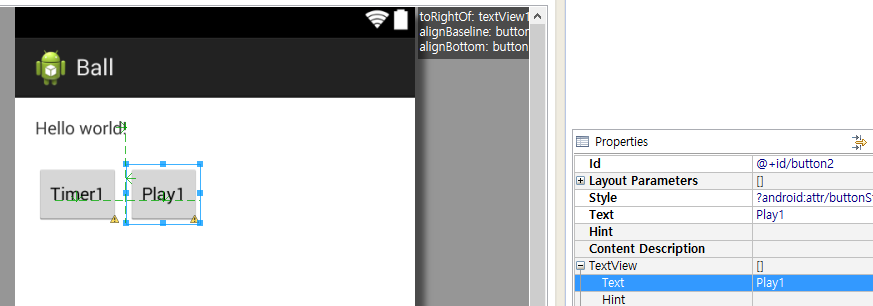
넷째 파라미터= 원 내부를 어떻게 칠할지 설정하기 위한 Paint 객체

# MainActivity 수정

게임 화면인 Play1Activity로 넘어가기 위한 버튼을 MainActivity에 추가하자.

## activity\_main.xml 수정

MainActivity의 화면 레이아웃인 res/layout/activity\_main.xml 파일을 열어서 버튼을 하나 추가하자.



Timer1Activity를 호출하기 위한 버튼을 하나 추가하자.

## MainActivity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | package skhu.sw.ball;  import android.support.v7.app.ActionBarActivity;  import android.content.Intent;  import android.os.Bundle;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.view.View.OnClickListener;  import android.widget.Button;  public class MainActivity extends ActionBarActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  Button button1 = (Button)findViewById(R.id.button1);  button1.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View v) {  Intent intent = new Intent(MainActivity.this, Timer1Activity.class);  startActivity(intent);  }  });  Button button2 = (Button)findViewById(R.id.button2);  button2.setOnClickListener(new OnClickListener() {  @Override  public void onClick(View v) {  Intent intent = new Intent(MainActivity.this, Play1Activity.class);  startActivity(intent);  }  });  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }  } |

(줄27~34) Play1Activity 액티비티 호출 기능 추가하였다.

# 실행

Play1Activity를 실행하여 볼의 움직임을 확인해 보자.

# Play2Activity 생성

공을 튕길 바(bar)를 구현하자.

게임 화면을 구현하자.

메뉴: File – New – Other – Android - Android Activity

Activity Name: Play2Activity

Hierarchical Parent: skhu.sw.ball.MainActivity

## Play2Activity.java 수정

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167 | package skhu.sw.ball;  import android.support.v7.app.ActionBarActivity;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.graphics.Paint;  import android.graphics.RadialGradient;  import android.graphics.Shader.TileMode;  import android.os.Bundle;  import android.os.Handler;  import android.os.Message;  import android.view.Menu;  import android.view.MenuItem;  import android.view.MotionEvent;  import android.view.View;  public class Play2Activity extends ActionBarActivity {    MyView2 myView;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  myView = new MyView2(this);  setContentView(myView);  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.play1, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  int id = item.getItemId();  if (id == R.id.action\_settings) {  return true;  }  return super.onOptionsItemSelected(item);  }      @Override  protected void onResume() {  super.onResume();  myView.startPlay();  }  @Override  protected void onPause() {  super.onPause();  myView.stopPlay();  }    }  class MyView2 extends View {    static final int TIMER\_INTERVAL = 15;  Handler handler;  Ball ball;  Bar bar;  boolean gameover;    public MyView2(Context context) {  super(context);  ball = new Ball();  bar = new Bar();  }    public void startPlay() {  handler = new Handler() {  public void handleMessage(Message msg) {  int width = getWidth(), height = getHeight() - 100;  ball.move(width, height);  if (bar.y == 0) {  bar.width = width / 2;  bar.move(width / 2, height);  }  invalidate();  if (!gameover)  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);  }  };  handler.sendEmptyMessageDelayed(0, TIMER\_INTERVAL);  }    public void stopPlay() {  handler.removeCallbacksAndMessages(null);  handler = null;  }    public void onDraw(Canvas canvas) {  ball.draw(canvas);  bar.draw(canvas);  }    public boolean onTouchEvent(MotionEvent event) {  int x = (int)event.getX();  int y = getHeight() - 100;  bar.move(x, y);  return true;  }    class Ball {  static final int RADIUS = 20;  int x, y;  int vx = 8, vy = 8;    public void move(int screenWidth, int screenHeight) {  x = x + vx;  y = y + vy;    if (x < 0) {  x = 0 - x;  vx = Math.abs(vx);  } else if (ball.x >= screenWidth) {  x = screenWidth - (x - screenWidth);  vx = -Math.abs(vx);  }  if (y < 0) {  y = 0 - y;  vy = Math.abs(vy);  } else if (ball.y >= screenHeight) {  if (bar.hit(ball)) {  y = screenHeight - (y - screenHeight);  vy = -Math.abs(vy);  } else  gameover = true;  }  }    void draw(Canvas canvas) {  Paint paint = new Paint();  RadialGradient gradient = new RadialGradient((x - RADIUS / 3), (y - RADIUS / 3), RADIUS, Color.WHITE, Color.BLUE, TileMode.CLAMP);  paint.setShader(gradient);  canvas.drawCircle(x, y, RADIUS, paint);  }    }    class Bar {  static final int HEIGHT = 20;  int x1, x2, y, width;    void move(int x, int y) {  x1 = x - width / 2;  x2 = x + width / 2;  this.y = y;  }    boolean hit(Ball ball) {  return x1 <= ball.x && ball.x <= x2;  }    void draw(Canvas canvas) {  Paint paint = new Paint();  paint.setColor(Color.BLACK);  canvas.drawRect(x1, y, x2, y + HEIGHT, paint);  }  }    } |

(줄65) Bar 멤버 변수를 추가되었다

(줄66) 볼을 바(bar)로 튕겨내는데 실패하면 gameover 변수가 true가 된다.

(줄70~71) 볼 객체와 바 객체를 생성한다

(줄77) 볼이 움직일 영역의 크기를 계산한다.

볼이 움직일 영역의 폭(width)은 화면 전체이다. 높이(height)는 바를 그릴 영역을 빼서 100만큼 줄였다.

(줄78) 볼은 파라미터로 주어진 영역 안에서 조금 이동한다.

(줄79) 바 객체의 멤버 변수를 아직 초기화 하지 않았다면, y 멤버 변수 값이 0일 것이다.

(줄80) 바의 폭을 화면의 폭의 반으로 설정한다

(줄81) 바의 위치를 화면 가운데 아래로 지정한다.

(줄84~85) 게임오바라면 더 이상 볼이 이동하지 않아야 하므로, 타이머 기능을 정지한다.

핸들러 객체에 메시지를 보내지 않으면, 줄76의 handleMessage 메소드가 더 이상 호출되지 않는다.

(줄101) 뷰(View) 객체가 터치 되면 이 메소드가 저절로 호출된다.

(줄102) 터치된 X 좌표를 구한다.

(줄103) 바(bar)가 그려질 Y 좌표를 구한다. 줄77에서 볼이 움직일 영역의 높이를 (화면 높이 – 100) 으로 계산했으므로, 바가 그려질 Y 좌표도 (화면 높이 – 100)으로 계산한다.

(줄104) 계산한 X, Y 좌표로 바를 이동한다.

(줄146~165) 바(bar)를 구현한 클래스이다.

(줄150) 주어진 좌표로 바의 좌표를 이동한다.

(줄156) 볼과 바가 만났는지를 x좌표로만 검사한다. 줄127~128에서 볼이 바 영역으로 내려갔을 때만 볼과 바가 만났는지 검사하므로, 볼의 y좌표는 다시 검사하지 않는다.

(줄160) 바 객체를 화면에 그린다.