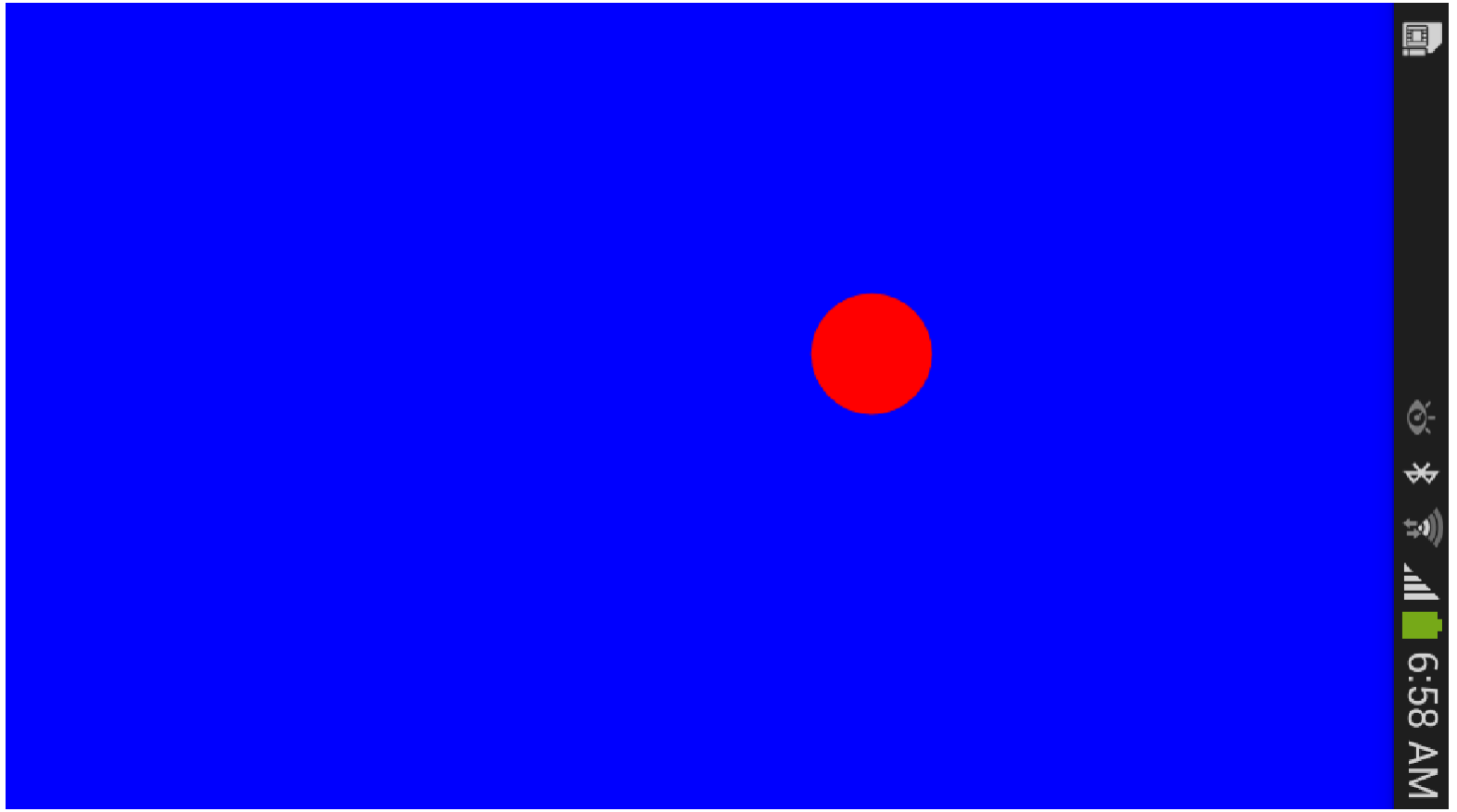
Bouncy Ball App for Android



Add WorldView in activity_main.xml

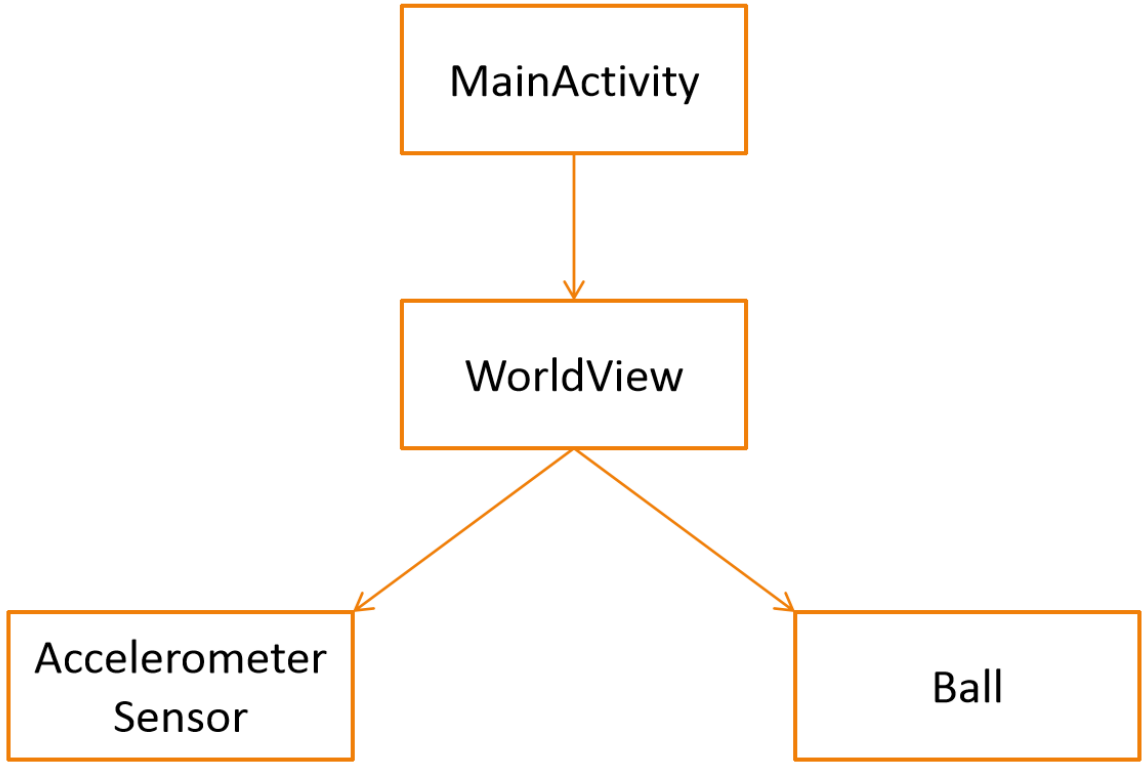
FrameLayout: designed to block out an area on the screen to display a single item.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:screenOrientation="portrait">

        <com.unimelb.bouncyball.WorldView
            android:id="@+id/worldView"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:screenOrientation="portrait" />

</FrameLayout>
```

Bouncy Ball Architecture



SensorManager and Sensor

- Define a SensorManager that allows you to access the device's sensors
- Get an instance of SensorManager class by calling Context.getSystemService() with the argument SENSOR_SERVICE
- Define a motion sensor that monitors the movement of a device
- Call registerListener() method to register a SensorEventListener for the given sensor at the given sampling frequency

```
private SensorManager mSensorManager;
private Sensor mAccelerometer;

mSensorManager = (SensorManager) context.getSystemService(context.SENSOR_SERVICE);
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

SensorEventListener

- Used for receiving notifications from the SensorManager when there is new sensor data
- Void onSensorChanged(SensorEvent event) called there is a new sensor event

```
mSensorManager.registerListener(new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        int accuracy = sensorEvent.accuracy;
        long timestamp = sensorEvent.timestamp;
        float values[] = sensorEvent.values;

        try {
            long curTime = System.currentTimeMillis();

            if ((curTime - lastUpdate) > 100) {
                lastUpdate = curTime;
                worldView.ball.setXSpeed(worldView.ball.getXSpeed()+((-1*values[0])/30));
                worldView.ball.setYSpeed(worldView.ball.getYSpeed()+(values[1]/30));
                System.out.println("@ X:"+values[0]+" Y:"+values[1]);
            }
        } catch(Exception e) {
        }
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {
    }
}, mAccelerometer, SensorManager.SENSOR_DELAY_GAME);
```

Ball Class

- Ball attribute: speed in X and Y direction, and position in X and Y direction
- moveBall(): move the ball

```
public void moveBall() {
    x = x + xSpeed;
    y = y + ySpeed;
}
```

Ball Class

- updatePhysics(): if the ball hit the border of your screen, the speed will be reverse

```
public void updatePhysics() {
    if(x > screenWidth-ballRadius) {
        //Reverse direction and slow down ball
        setXSpeed(getXSpeed()*-1);
    }
    if(x < ballRadius) {
        //Reverse direction and slow down ball
        setXSpeed(getXSpeed()*-1);
    }
    if(y > screenHeight-ballRadius) {
        //Reverse direction and slow down ball
        setYSpeed(getYSpeed()*-1);
    }
    if(y < ballRadius) {
        if(worldView.connected == false) {
            //Reverse direction and slow down ball
            setYSpeed(getYSpeed()*-1);
        }
        else {
            if(worldView.onScreen == true) {
                sendBluetoothMessage();
                //Send a message to connected phone to show ball
            }
        }
    }
```

Ball Class

```
public void sendBluetoothMessage() {
    try {
        StringBuffer sb = new StringBuffer();
        sb.append("ShowOnScreen");
        sb.append(",");
        sb.append(String.valueOf(screenWidth));
        sb.append(",");
        sb.append(String.valueOf(screenHeight));
        sb.append(",");
        sb.append(String.valueOf(x));
        sb.append(",");
        sb.append(String.valueOf(y));
        sb.append(",");
        sb.append(String.valueOf(xSpeed));
        sb.append(",");
        sb.append(String.valueOf(ySpeed));
        sb.append("\n");
        worldView.onScreen = false;
        worldView.outputStream.write(sb.toString().getBytes());
        worldView.outputStream.flush();
    } catch(Exception e) {
    }
}
```

Ball Class

- onDraw(Canvas): get called to draw into worldview

```
public void onDraw(Canvas canvas) {
    Paint paint = new Paint();
    //smooth out the edges of what is being draw
    paint.setAntiAlias(true);
    paint.setColor(Color.RED);

    updatePhysics();

    if(worldView.onScreen) {
        moveBall();
        canvas.drawCircle(x, y, ballRadius, paint);
    }
}
```

WorldView extends SurfaceView

- surfaceCreated(SurfaceHolder surfaceHolder): called immediately after the surface is first created
- SurfaceHolder interface: allows you to control the surface size and format, edit the pixels in the surface, and monitor the changes to the surface

```
@Override
public void surfaceCreated(SurfaceHolder surfaceHolder) {
    this.surfaceHolder = surfaceHolder;
    this.running = true;

    width = getWidth();
    height = getHeight();
    ball = new Ball(this, null, width, height);

    Thread t = new Thread(this);
    t.start();
}
```

A thread to constantly draw the ball

- Canvas lockCanvas(): the returned canvas can be used to draw into the surface's bitmap
- unlockCanvasAndPost(): finish editing in the surface, the surface's current pixels will be displayed on the screen

```
public void run() {
    while(running) {
        Canvas canvas = null;
        try {
            canvas = surfaceHolder.lockCanvas(null);
            onDraw(canvas);
            ball.onDraw(canvas);
        } finally {
            if (canvas != null) {
                surfaceHolder.unlockCanvasAndPost(canvas);
            }
        }
        try {
            Thread.sleep(10);
        } catch(Exception e) {}
    }
}
```

MainActivity

```
public class MainActivity extends Activity {
    private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"); //UUID for Bluetooth Connections
    private final static int REQUEST_ENABLE_BT = 1;

    private BluetoothAdapter mBluetoothAdapter;
    private ServerThread myServer;
    private boolean isServer = false;
    private WorldView worldView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT); //Force device to stay in portrait orientation
        requestWindowFeature(Window.FEATURE_NO_TITLE); //Remove banner from the top of the activity
        setContentView(R.layout.activity_main); //Set the layout to activity_main

        worldView = (WorldView) findViewById(R.id.worldView);
        startBluetooth();
    }
```