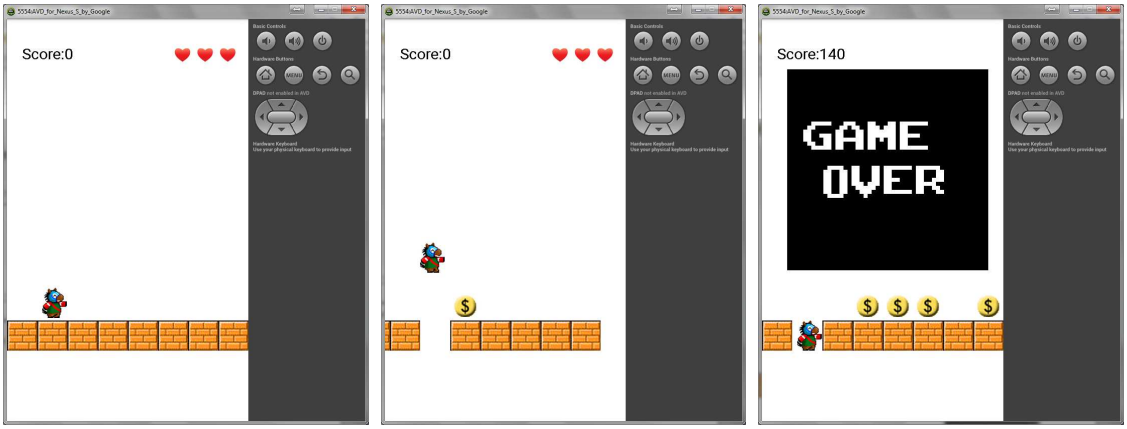
Workshop

1. Create a simple Canvas view with simple drawing *(Page 1 – 5*)
   - Hide Action Bar and Status Bar
   - Create Canvas View
   - Create Simple Drawing

2. Create a simple animation (*Page 6 – 10*)
   - Load a simple image
   - Create a simple animation

3. Create a simple paint brush that allows user to draw something (*Page 11 – 13*)

4. **Exercise:** Now it's time for you to develop a simple game. The hero needs to walk along a long path. The score will be increased once the hero gets a coin. However, one life will be deduced when it drops to the hole. The game will be over when all life are lost. User can press the screen to trigger the hero to jump over the hole.



1. Using Canvas

1.1 Hide Action Bar and Status Bar

1. Create the Android application with the following attributes.
   - Application Name: **MyDrawing**
   - Project Name: **MyDrawing**
   - Package Name: **com.example.mydrawing**

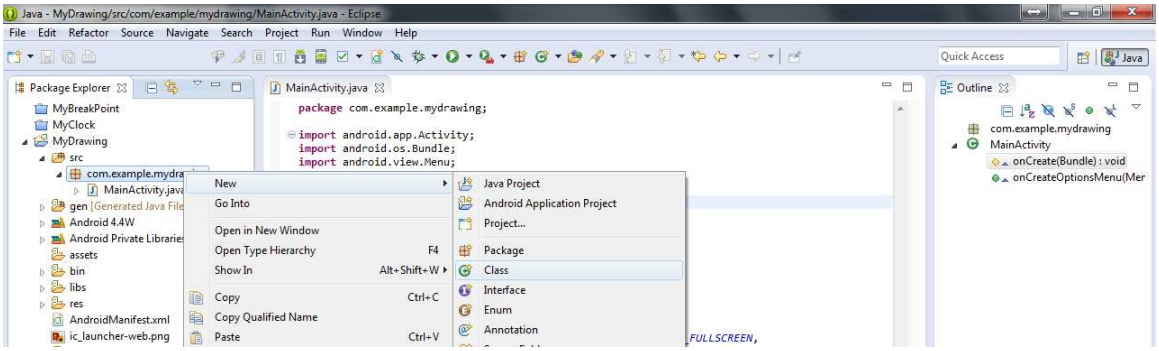
2. Modify the source code of "**MainActivity.java**" as follow.

```
package com.example.mydrawing;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.Window;
import android.view.WindowManager;

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

        // Hide Action Bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        // Hide status bar
        getWindow().setFlags( WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

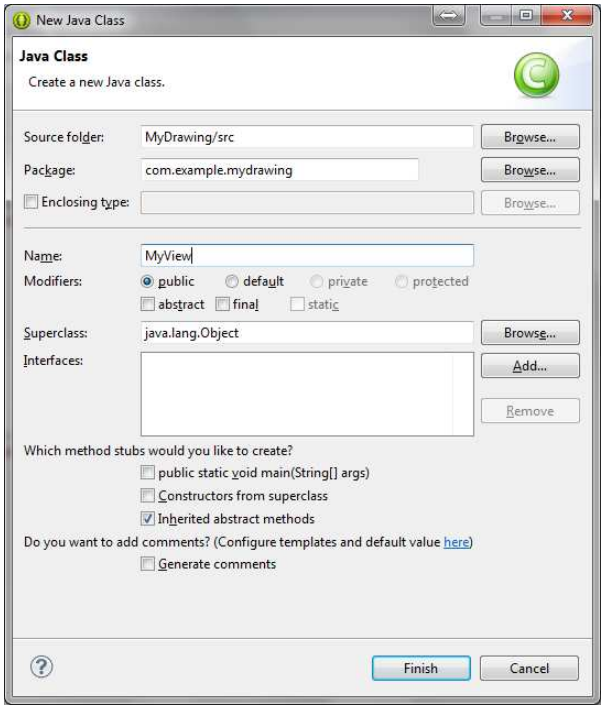
3. Save and execute the apps**,** you action bar and status bar should be hide

1.2 Create Custom View

1. Open the previous project, Right click the package **com.example.mydrawing**, and select **New ➔ Class**.



2. Input "**MyView**" as the class name and press **[Finish]** button.



3. Modify the source code of "**MyView.java**" as follow.

```
package com.example.mydrawing;

import android.content.Context;
import android.view.View;

public class MyView extends View {
    public MyView(Context context) {
        super(context);
    }
}
```

4. Modify the source code of "**MainActivity.java**" as follow.

```
package com.example.mydrawing;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.Window;
import android.view.WindowManager;
import android.view.View;

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

        // Hide Action Bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        // Hide status bar
        getWindow().setFlags( WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);

//       setContentView(R.layout.activity_main);
         View MyView = new MyView(this);
         setContentView(MyView);
    }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

4. Save and execute the apps**,** can you see a blank screen?

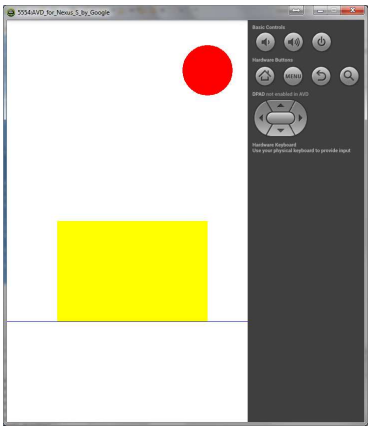
1.3 Simple Drawing

1. Modify the source code of "**MyView.java**" as follow.

```
package com.example.mydrawing;

import android.content.Context;
import android.view.View;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;

public class MyView extends View {
    public Paint mPaint;  // The paint used for drawing

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

        // Construct the paint (e.g. style, color)
        mPaint = new Paint();
    }
```

```
    @Override
    public void onDraw(Canvas canvas) {
        // Draw a red circle at (400, 100) with radius 50
        mPaint.setColor(Color.RED);
        canvas.drawCircle(400, 100, 50, mPaint);

        // Draw a yellow rectangle from (100, 400) to (400, 600)
        mPaint.setColor(Color.YELLOW);
        canvas.drawRect(100, 400, 400, 600, mPaint);

        // Draw a blue lines from (0, 600) to (480, 600)
        mPaint.setColor(Color.BLUE);
        canvas.drawLine(0, 600, 480, 600, mPaint);
    }
}
```

2. Save and execute the apps, can you see your beautiful drawing?



2. Simple Animation

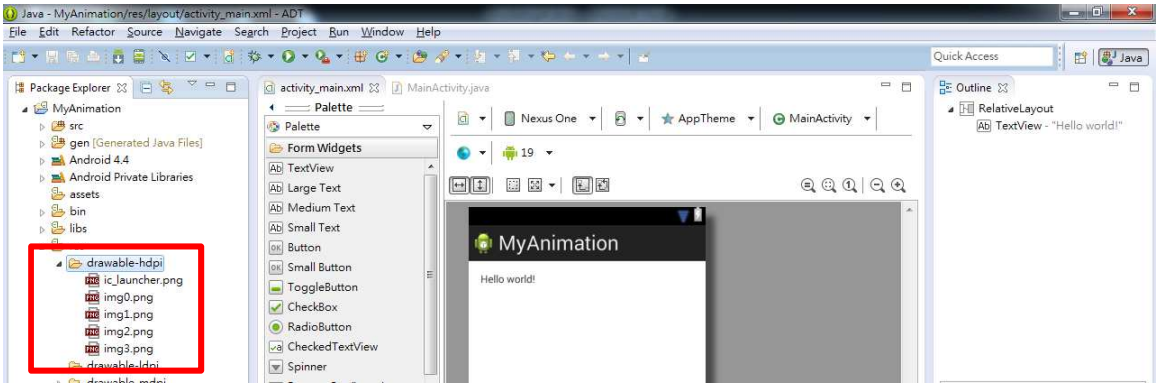
2.1 Load Simple Picture

1. Create the Android application with the following attributes.
   - Application Name: **MyAnimation**
   - Project Name: **MyAnimation**
   - Package Name: **com.example.myianimation**

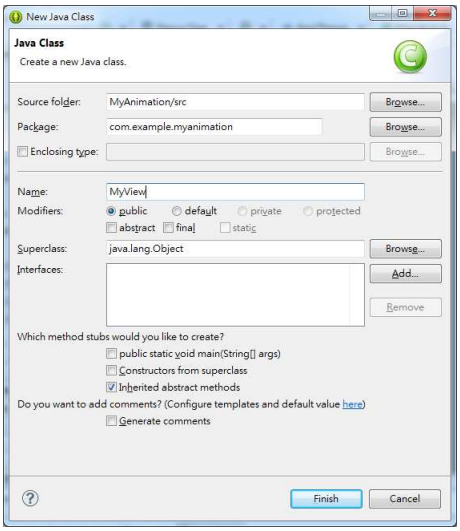
2. Name the pictures in a sequential order.

img0.png img1.png img2.png img3.png

3. Put the images into the folder **drawable-hdpi** (**drawable-xxhdpi** for high resolution model).



4. Right click the package **com.example.myanimation**, and select **New ➔ Class**. Input "**MyView**" as the class name and press **[Finish]** button.



5. Modify the source code of "**MyView.java**" as follow.

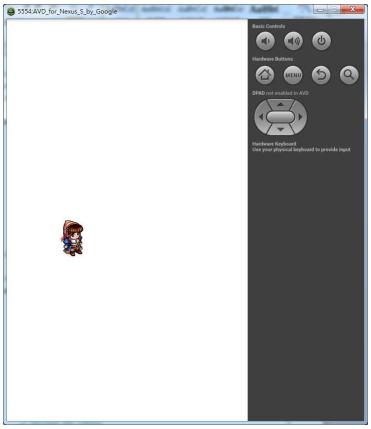
```
package com.example.myanimation;

import android.content.Context;
import android.view.View;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;

public class MyView extends View {
    public int N = 0;
    public Bitmap[] mImage;

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

        // Construct the Bitmap array
        mImage = new Bitmap[10];

        // Read the image from Drawable folder
        mImage[0] = BitmapFactory.decodeResource(getResources(), R.drawable.img0);
    }

    @Override
    public void onDraw(Canvas canvas) {
        // Display the image at (100, 400)
        canvas.drawBitmap(mImage[N], 100, 400, null);
    }
}
```

6. Modify the source code of "**MainActivity.java**" as follow.

```
package com.example.myanimation;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.Window;
import android.view.WindowManager;
import android.view.View;
```

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

       // Hide Action Bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);

       // Hide status bar
        getWindow().setFlags( WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);

//       setContentView(R.layout.activity_main);
         View MyView = new MyView(this);
         setContentView(MyView);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

7. Save and execute the apps, can you see the lovely girl?



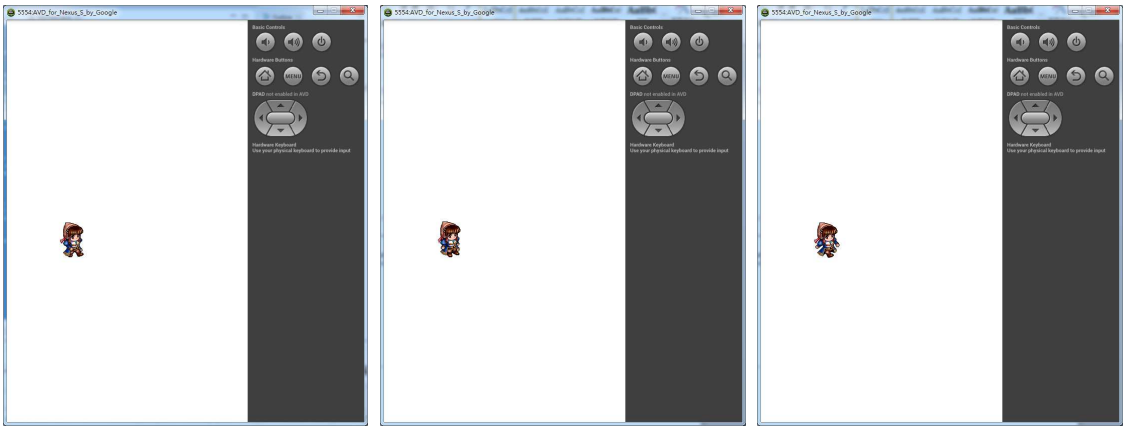
2.2 Create Simple Animation

1. Modify the source code of "**MyView.java**" as follow.

```
package com.example.myanimation;

import android.content.Context;
import android.view.View;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;

public class MyView extends View {
    public int N = 0;
    public Bitmap[] mImage;

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

        // Construct the Bitmap array
        mImage = new Bitmap[10];

        // Read the image from Drawable folder
        mImage[0] = BitmapFactory.decodeResource(getResources(), R.drawable.img0);
        mImage[1] = BitmapFactory.decodeResource(getResources(), R.drawable.img1);
        mImage[2] = BitmapFactory.decodeResource(getResources(), R.drawable.img2);
        mImage[3] = BitmapFactory.decodeResource(getResources(), R.drawable.img3);
    }

    @Override
    public void onDraw(Canvas canvas) {
        // Display the image at (100, 400)
        canvas.drawBitmap(mImage[N], 100, 400, null);

        // Sleep for 60ms
        try {
            Thread.sleep(60);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            // Increase the counter
            if (N < 3) {
                N++;
            } else {
                N = 0;
            }

            // Redraw the screen
            invalidate();
        }
}
```

2. Save and execute the apps, can you see the animation?

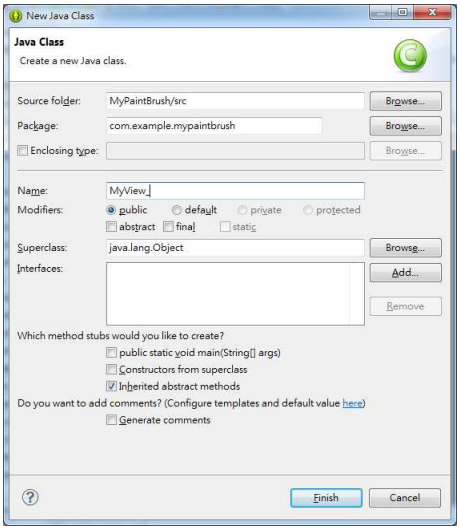


3. Paint Brush

3.1 Handle Touch Event

1. Create the Android application with the following attributes.
   - Application Name: **MyPaintBrush**
   - Project Name: **MyPaintBrush**
   - Package Name: **com.example.mypaintbrush**

2. Right click the package **com.example.mypaintbrush**, and select **New ➔ Class**. Input "**MyView**" as the class name and press **[Finish]** button.



3. Modify the source code of "**MyView.java**" as follow.

```
package com.example.mypaintbrush;

import android.content.Context;
import android.view.View;
import android.view.MotionEvent;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import java.util.ArrayList;
import java.util.List;

public class MyView extends View {
    List<Point> points = new ArrayList<Point>();
    Paint paint = new Paint();
```

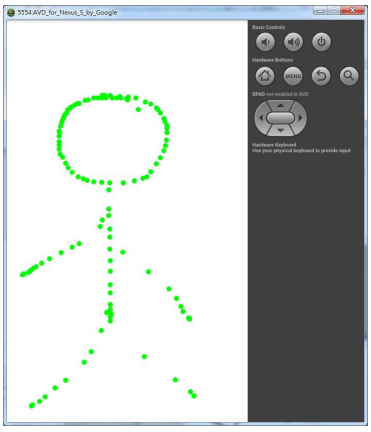
```
    public MyView(Context context) {
        super(context);
        paint.setColor(Color.GREEN);
        paint.setAntiAlias(true);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        Point point = new Point();

        // Obtain the coordinate for the touch point
        point.x = (int)event.getX();
        point.y = (int)event.getY();

        // Add the point to the list
        points.add(point);

        // Redraw the screen
        invalidate();
        return true;
    }

    @Override
    public void onDraw(Canvas canvas) {
        for (Point point : points) {
            // Draw the point on screen
            canvas.drawCircle(point.x, point.y, 5, paint);
        }
    }
}
```

4. Modify the source code of "**MainActivity.java**" as follow.

```
package com.example.mypaintbrush;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.Window;
```

```
import android.view.WindowManager;
import android.view.View;

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

        // Hide Action Bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        // Hide status bar
        getWindow().setFlags( WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);

//       setContentView(R.layout.activity_main);
         View MyView = new MyView(this);
         setContentView(MyView);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

5. Save and execute the apps, now you can draw something on your screen.



4. Game Development

4.1 Draw the Initial Game World

1. Create the Android application with the following attributes.
   - Application Name: **MyMotionGame**
   - Project Name: **MyMotionGame**
   - Package Name: **com.example.mymotiongame**

2. Put the images into the folder **res/drawable-xxhdpi**.


brick.png


life.png


hero0.png


hero1.png


dollar.png

gameover.png

3. Right click the package **com.example.mymotiongame**, and select **New ➔ Class**. Input "**MyView**" as the class name and press **[Finish]** button. Modify the source code of "**MyView.java**" as follow.

```
package com.example.mymotiongame;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;

public class MyView extends View {
    // Define the constant for game status
    final int GAME_PLAY = 1,            // Game Status: Playing
              GAME_DROP = 2,            // Game Status: Drop to Hole
              GAME_OVER = 9;            // Game Status: Game Over

    private Paint mPaint;               // The paint used for drawing
    private Bitmap[] imgHero;           // Graphic for Hero moving
    private Bitmap imgBrick;            // Graphic for Brick
    private Bitmap imgLife;             // Graphic for Life
    private Bitmap imgDollar;           // Graphic for Dollar
```

```
    private Bitmap imgGameOver;         // Graphic for GameOver
    private int HeroPicture = 0;        // Index for Hero picture
    private int Map[] = new int[10];    // Map
    private int CurrentLife = 3;        // Number of Life leave
    private int score = 0;              // Display the Score
    private int HeroY = 535;            // Y-coordinate for Hero
    private int MapX = 0;               // X coordinate for Map
    private int ScreenMode = 1;         // Game Screen Status

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

        // Set the font size and color
        mPaint = new Paint();
        mPaint.setColor(Color.BLACK);
        mPaint.setTextSize(30);

        // Construct the Bitmap array
        imgHero = new Bitmap[2];

        // Read the image from Drawable folder
        imgBrick = BitmapFactory.decodeResource(getResources(), R.drawable.brick);
        imgLife = BitmapFactory.decodeResource(getResources(), R.drawable.life);
        imgDollar = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.dollar);
        imgGameOver = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.gameover);
        imgHero[0] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero0);
        imgHero[1] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero1);

        // Initial the map (set it as ground first)
        for (int i=0; i<9; i++) {
            Map[i] = 2;
        }
    }

    @Override
```

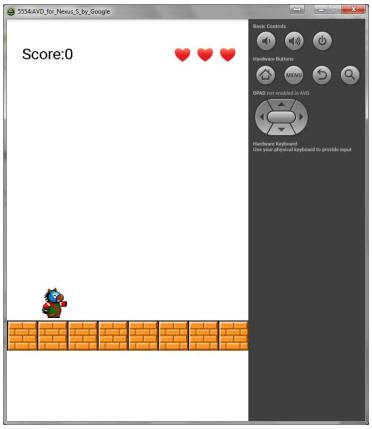
```
    public void onDraw(Canvas canvas) {
        // Draw the game screen
        for (int i=0; i<9; i++) {
            switch (Map[i]) {
                case 0:  // 20% to display a hole
                    break;
                case 1:  // 20% to display a coin
                    canvas.drawBitmap(imgDollar, MapX + 60*i, 550, null);
                    canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                    break;
                default: // default to display a ground
                    canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                    break;
            }
        }
        // Display the number of life
        for (int i=1; i<=CurrentLife; i++) {
            canvas.drawBitmap(imgLife, 280 + 45*i, 50, null);
        }

        // Draw the score
        canvas.drawText("Score:" + score, 30, 80, mPaint);

        // Draw the Hero
        canvas.drawBitmap(imgHero[HeroPicture], 70, HeroY, null);

        // Draw the Game Over screen
        if (ScreenMode == GAME_OVER ) {
            canvas.drawBitmap(imgGameOver, 50, 100, null);
        }
    }
}
```

4. Modify the source code of "**MainActivity.java**" as follow.

```
package com.example.mymotiongame;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.Window;
import android.view.WindowManager;
import android.view.View;

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

        // Hide Action Bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        // Hide status bar
        getWindow().setFlags( WindowManager.LayoutParams.FLAG_FULLSCREEN,
                              WindowManager.LayoutParams.FLAG_FULLSCREEN);

//       setContentView(R.layout.activity_main);
         View MyView = new MyView(this);
         setContentView(MyView);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

5. Save and execute the app, you should able to see the initial game world.



4.2 Handle the Character Movement

1. Open the previous project and modify the source code of "**MyView.java**" as follow.

```
package com.example.mymotiongame;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;
import java.util.Random;

public class MyView extends View {
    // Define the constant for game status
    final int GAME_PLAY = 1,            // Game Status: Playing
              GAME_DROP = 2,            // Game Status: Drop to Hole
              GAME_OVER = 9;            // Game Status: Game Over

    private Paint mPaint;               // The paint used for drawing
    private Bitmap[] imgHero;           // Graphic for Hero moving
    private Bitmap imgBrick;            // Graphic for Brick
    private Bitmap imgLife;             // Graphic for Life
    private Bitmap imgDollar;           // Graphic for Dollar
    private Bitmap imgGameOver;         // Graphic for GameOver
    private int HeroPicture = 0;        // Index for Hero picture
    private int Map[] = new int[10];    // Map
    private int CurrentLife = 3;        // Number of Life leave
    private int score = 0;              // Display the Score
    private int HeroY = 535;            // Y-coordinate for Hero
    private int MapX = 0;               // X coordinate for Map
    private int ScreenMode = 1;         // Game Screen Status
    private int HeroSpeed = 10;         // The Walking speed

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

        // Set the font size and color
        mPaint = new Paint();
```

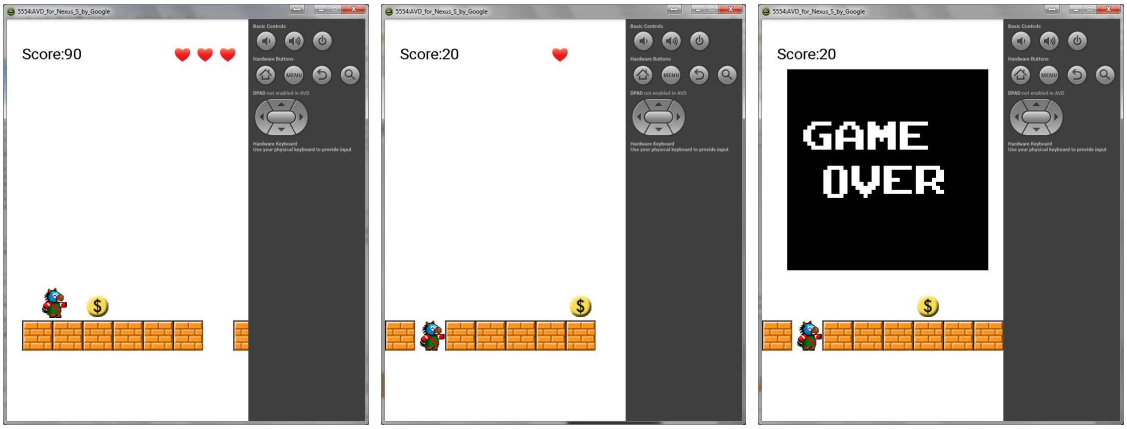
```
        mPaint.setColor(Color.BLACK);
        mPaint.setTextSize(30);

        // Construct the Bitmap array
        imgHero = new Bitmap[2];

        // Read the image from Drawable folder
        imgBrick = BitmapFactory.decodeResource(getResources(), R.drawable.brick);
        imgLife = BitmapFactory.decodeResource(getResources(), R.drawable.life);
        imgDollar = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.dollar);
        imgGameOver = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.gameover);
        imgHero[0] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero0);
        imgHero[1] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero1);

        // Initial the map (set it as ground first)
        for (int i=0; i<9; i++) {
            Map[i] = 2;
        }
    }

    @Override
    public void onDraw(Canvas canvas) {
        // Generate the Random Number when program start
        Random randomize = new Random();

        // Move the map to left according to Hero speed
        MapX = MapX - HeroSpeed;

        if (MapX < -59) {
            // Shift the Map to left
            MapX = 0;
            for (int i=0; i<9; i++) {
                Map[i] = Map[i+1];
            }
            // Generate the next map
```

```
            Map[9] = randomize.nextInt(5);
        }

        // Set the Hero to Walk if current stand
        if (HeroPicture == 1) {
            HeroPicture = 0;
        } else {
            HeroPicture = 1;
        }

        // Handle hero drop to hole
        if (ScreenMode == GAME_DROP) {
            try {
                Thread.sleep(2000);        // Stop the screen for 2 second
            } catch (Exception e) {
                e.printStackTrace();
            }

            // Reset the Screen
            for (int i=0; i<9; i++) {  // Reset the map to ground
                Map[i] = 2;
            }
            HeroY = 535;               // Reset Hero location
            ScreenMode = GAME_PLAY;    // Reset the Game Screen Flag
        }

        // Check for Get Coin action
        if (Map[2] == 1 && HeroY > 500) {
            score = score + 10;        // Increase the score if get a coin
            Map[2] = 2;                // Reset the map to ground
        }

        // Check for Drop to a hole
        if (Map[1] == 0 && HeroY > 500) {
            HeroY = 600;                    // Draw the hero in the hole
            CurrentLife = CurrentLife - 1; // Reduce one life
            if ( CurrentLife <= 0 ) {
                ScreenMode = GAME_OVER;    // Change the flag to GAME_OVER
            } else {
```

```
                ScreenMode = GAME_DROP;    // Change the flag to GAME_DROP
            }
        }

        // Draw the game screen
        for (int i=0; i<9; i++) {
            switch (Map[i]) {
                case 0:  // 20% to display a hole
                    break;
                case 1:  // 20% to display a coin
                    canvas.drawBitmap(imgDollar, MapX + 60*i, 550, null);
                    canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                    break;
                default: // default to display a ground
                    canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                    break;
            }
        }

        // Display the number of life
        for (int i=1; i<=CurrentLife; i++) {
            canvas.drawBitmap(imgLife, 280 + 45*i, 50, null);
        }

        // Draw the score
        canvas.drawText("Score:" + score, 30, 80, mPaint);

        // Draw the Hero
        canvas.drawBitmap(imgHero[HeroPicture], 70, HeroY, null);

        // Draw the Game Over screen
        if (ScreenMode == GAME_OVER ) {
            canvas.drawBitmap(imgGameOver, 50, 100, null);
        }

        try {
            if ( ScreenMode != GAME_OVER ) {
                Thread.sleep(60);          // Sleep for 60ms
                invalidate();              // Redraw the screen
```

```
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

2. Save and execute the app, the score can be increased when you get a coin. However, the hero dies easily because it always drops to the hole.



4.3 Handle the Touch Event (for Hero Jump)

1. Open the previous project and modify the source code of "**MyView.java**" as follow.

```
package com.example.mymotiongame;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.View;
import java.util.Random;
import android.view.MotionEvent;

public class MyView extends View {
    // Define the constant for game status
    final int GAME_PLAY = 1,            // Game Status: Playing
              GAME_DROP = 2,            // Game Status: Drop to Hole
              GAME_OVER = 9;            // Game Status: Game Over
```

```
    private Paint mPaint;               // The paint used for drawing
    private Bitmap[] imgHero;           // Graphic for Hero moving
    private Bitmap imgBrick;            // Graphic for Brick
    private Bitmap imgLife;             // Graphic for Life
    private Bitmap imgDollar;           // Graphic for Dollar
    private Bitmap imgGameOver;         // Graphic for GameOver
    private int HeroPicture = 0;        // Index for Hero picture
    private int Map[] = new int[10];    // Map
    private int CurrentLife = 3;        // Number of Life leave
    private int score = 0;              // Display the Score
    private int HeroY = 535;            // Y-coordinate for Hero
    private int MapX = 0;               // X coordinate for Map
    private int ScreenMode = 1;         // Game Screen Status
    private int HeroSpeed = 10;         // The Walking speed
    private int JumpAction = 0;         // Jump Action

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

        // Set the font size and color
        mPaint = new Paint();
        mPaint.setColor(Color.BLACK);
        mPaint.setTextSize(30);

        // Construct the Bitmap array
        imgHero = new Bitmap[2];

        // Read the image from Drawable folder
        imgBrick = BitmapFactory.decodeResource(getResources(), R.drawable.brick);
        imgLife = BitmapFactory.decodeResource(getResources(), R.drawable.life);
        imgDollar = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.dollar);
        imgGameOver = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.gameover);
        imgHero[0] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero0);
        imgHero[1] = BitmapFactory.decodeResource(getResources(),
                                                 R.drawable.hero1);
```

```
        // Initial the map (set it as ground first)
        for (int i=0; i<9; i++) {
            Map[i] = 2;
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        JumpAction = 1;
        return true;
    }

    @Override
    public void onDraw(Canvas canvas) {
        // Generate the Random Number when program start
        Random randomize = new Random();

        // Move the map to left according to Hero speed
        MapX = MapX - HeroSpeed;

        if (MapX < -59) {
            // Shift the Map to left
            MapX = 0;
            for (int i=0; i<9; i++) {
                Map[i] = Map[i+1];
            }
            // Generate the next map
            Map[9] = randomize.nextInt(5);
        }

        // Set the Hero to Walk if current stand
        if (HeroPicture == 1) {
            HeroPicture = 0;
        } else {
            HeroPicture = 1;
        }

        // Handle the Jump up action
```

```
        if (JumpAction == 1) {
            if (HeroY > 400){
                HeroY = HeroY - 30;    // Jump up until a defined high
            } else {
                JumpAction = -1;       // Reset the Jump Directions to Down
            }
            // Handle the Jump Down action
        } else if (JumpAction == -1) {
            if (HeroY < 500) {
                HeroY = HeroY + 30;    // Free Fall to the ground
            } else {
                HeroY = 535;
                JumpAction = 0;        // Reset to walking mode
            }
        }

        // Handle hero drop to hole
        if (ScreenMode == GAME_DROP) {
            try {
                Thread.sleep(2000);        // Stop the screen for 2 second
            } catch (Exception e) {
                e.printStackTrace();
            }

            // Reset the Screen
            for (int i=0; i<9; i++) {  // Reset the map to ground
                Map[i] = 2;
            }
            HeroY = 535;               // Reset Hero location
            JumpAction = 0;            // Set the Hero to Walk (No Jump)
            ScreenMode = GAME_PLAY;    // Reset the Game Screen Flag
        }

        // Check for Get Coin action
        if (Map[2] == 1 && HeroY > 500) {
            score = score + 10;        // Increase the score if get a coin
            Map[2] = 2;                // Reset the map to ground
        }
```

```
        // Check for Drop to a hole
        if (Map[1] == 0 && HeroY > 500) {
             HeroY = 600;                          // Draw the hero in the hole
             CurrentLife = CurrentLife - 1;        // Reduce one life
             if ( CurrentLife <= 0 ) {
                  ScreenMode = GAME_OVER;          // Change the flag to GAME_OVER
             } else {
                  ScreenMode = GAME_DROP;          // Change the flag to GAME_DROP
             }
        }

        // Draw the game screen
        for (int i=0; i<9; i++) {
             switch (Map[i]) {
                  case 0:  // 20% to display a hole
                       break;
                  case 1:  // 20% to display a coin
                       canvas.drawBitmap(imgDollar, MapX + 60*i, 550, null);
                       canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                       break;
                  default: // default to display a ground
                       canvas.drawBitmap(imgBrick, MapX + 60*i, 600, null);
                       break;
             }
        }

        // Display the number of life
        for (int i=1; i<=CurrentLife; i++) {
             canvas.drawBitmap(imgLife, 280 + 45*i, 50, null);
        }

        // Draw the score
        canvas.drawText("Score:" + score, 30, 80, mPaint);

        // Draw the Hero
        canvas.drawBitmap(imgHero[HeroPicture], 70, HeroY, null);

        // Draw the Game Over screen
        if (ScreenMode == GAME_OVER ) {
```

```
                canvas.drawBitmap(imgGameOver, 50, 100, null);
            }


            try {
                if ( ScreenMode != GAME_OVER ) {
                    Thread.sleep(60);          // Sleep for 60ms
                    invalidate();              // Redraw the screen
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
}
```

2. Save and execute the app, touch the screen to dump across the hole. How many marks can you get?