Мобайл програмчлал – Лекц 13

Ж.Золжаргал
(j.zoljargal@must.edu.mn)

Building Rich User Interfaces

- Гар утасны програмчлал хөгжихийн хэрээр UI чухал болж байна
- Энэ хичээлээр, Сүүдэрлэлт, тунгалаг байдал, хөдөлгөөн, touch screen зэргийг өөрийн Activity болон View –д хэрхэн ашиглах талаар үзнэ.

Working with Animations

- Андройдод animation гүйцэтгэх 2 арга байдаг
  - –Frame-by-frame animation уламжлалт арга бөгөөд frame бүрт ялгаатай агшин дүрслэгддэг. Энэ view дотор дүрслэгддэг бөгөөд Canvas-г ашигладаг
  - –Tweened animation нь view-д үйлчилдэг. Түүний өөрчлөгдөх байршил, хэмжээ, эргэлт зэргийг зааж өгнө.

Introducing Tweened Animations

- Tweened animation нь энгийн бөгөөд хамгийн багаар нөөцийг шаарддаг
- Чиглэл, байрлал, тунгалаг болгох гэх мэт зүйлсийг бид өөрсдөө Canvas ашиглаж гүйцэтгэсэнээс илүү бага нөөцийг ашигладаг

Introducing Tweened Animations

- Tweened animation нь дараах тохиолдолуудад өргөн хэрэглэгддэг
  - Activity хооронд шилжих
  - Activity дахь layout хооронд шилжих
  - Тухайн View дэхь агуулга хооронд шилжих
  - Хэрэглэгч рүү хариу үйлдэл үзүүлэх /feadback/
  - Элсэн цагаар процессийг дүрслэх
  - Input box-н өгөгдлийг зөв эсвэл буруу оруулсан тохиолдолд түүнийг “Сэгсрэх”

Creating Tweened Animations

- Tweened animation-г Animation классаар үүсгэх бөгөөд доорх ангиллуудтай
  - AlphaAnimation – view тунгалаг байх эсэхийг заана
  - RotateAnimation – сонгосон view XY хавтгайд эргэх
  - ScaleAnimation – тухайн view-г томруулах жижгэрүүлэхийг заана
  - TranslateAnimation – тухайн view-г зөөх

Creating Tweened Animations

- AnimationSet класс-д ажиллах animation-уудыг багцлаж тохиоруулна.
- Эхлэх цаг, animation бүрийн үргэлжлэх хугацаа болон дараалал зэргийг зааж өгнө.

- Ямар хугацааны дараа эхлэх болон animation бүрийн үргэлжлэх хугацааг зааж өгөх нь чухал, эсвэл бүгд зэрэг эхлээд зэрэг дуусах уу

Creating Tweened Animations

```
// Create the AnimationSet
AnimationSet myAnimation = new AnimationSet(true);

// Create a rotate animation.
RotateAnimation rotate = new RotateAnimation(0, 360,
                                             RotateAnimation.RELATIVE_TO_SELF,0.5f,
                                             RotateAnimation.RELATIVE_TO_SELF,0.5f
                                            );
rotate.setFillAfter(true);
rotate.setDuration(1000);

// Create a scale animation
ScaleAnimation scale = new ScaleAnimation(1, 0,
                                          1, 0,
                                          ScaleAnimation.RELATIVE_TO_SELF, 0.5f,
                                          ScaleAnimation.RELATIVE_TO_SELF, 0.5f
                                         );
scale.setFillAfter(true);
scale.setDuration(500);
scale.setStartOffset(500);
```

Creating Tweened Animations

```
// Create an alpha animation
AlphaAnimation alpha = new AlphaAnimation(1, 0);
scale.setFillAfter(true);
scale.setDuration(500);
scale.setStartOffset(500);

// Add each animation to the set
myAnimation.addAnimation(rotate);
myAnimation.addAnimation(scale);
myAnimation.addAnimation(alpha);
```

Applying Tweened Animations

- View-ийн startAnimation функцыг дуудсанаар идэвхжинэ
- Animation нь нэг л удаа ажиллаад зогсох ба дараах функцуудээр тохируулж болно
  - –setRepeatMode
  - –setRepeatCount

```
myAnimation.setRepeatMode(Animation.RESTART);
myAnimation.setRepeatCount(Animation.INFINITE);

myView.startAnimation(myAnimation);
```

Using Animation Listeners

- AnimationListener –н тусламжтай тухайн animation-ны event-үүдийг барьж авах боломжтой
- Ингэснээр animation эхлэх болон дуусах үзэгдлийг барих, View-н агуулгыг өөрчлөх, олон animation-г цувуулан тоглуулах гэх мэт

Using Animation Listeners

```
myAnimation.setAnimationListener(new AnimationListener() {

  public void onAnimationEnd(Animation _animation) {
    // TODO Do something after animation is complete.
  }


  public void onAnimationRepeat(Animation _animation) {

    // TODO Do something when the animation repeats.
  }

  public void onAnimationStart(Animation _animation) {
    // TODO Do something when the animation starts.
  }

});
```

Animated Sliding User Interface Example

- Жишээ ажиллуулж үзэх
  - –Чиглэл заасан товчлуур дээр дарахад View-ийн агуулгыг солиж, тухайн чиглэлд View-г хөдөлгөнө.

Animating Layouts and View Groups

- LayoutAnimation нь View Group элементүүдэд хэрэглэгддэг
- LayoutAnimationController-оор View Group-н View -н animation-г тодорхойлж өгнө
- Үүнд агуулагдах View бүр нь ижилхэн animation гүйцэтгэх бөгөөд тэдгээрийн дараалал болон эхлэх хугацааг зааж өгч болно

Animating Layouts and View Groups

- Андройд нь хоёр төрлийн LayoutAnimationController класс агуулдаг
  - LayoutAnimationController – View бүрийн эхлэх хугацааг (millisecond) зааж өгч болно, эсвэл дараахаас сонгож болно
    - Forward, reverse, random
  - GridLayoutAnimationController – энэ нь хоёрдогч класс бөгөөд animation идэвхжих дарааллыг мөр болон баганаар тодорхойлж болно

Creating and Using Frame-by-Frame Animations

- Frame-by-frame animation нь уламжлалт хүүхэлдэйн кино хийдэг загвар бөгөөд frame бүрт нэг зураг харгалзана
- Tweened animation-д View дээр animation хийдэг байсан бол Frame-by-frame animation-д цуваа Drawable объект тодорхойлох бөгөөд тэдгээр нь View-н дэвсгэр зураг болно.

Creating and Using Frame-by-Frame Animations

- AnimationDrawable класс нь Frame-by-frame animation үүсгэхэд хэрэглэгдэх бөгөөд Drawable нөөцүүдийг дүрслэн харуулна.

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
                android:oneshot="false">
  <item android:drawable="@drawable/rocket1" android:duration="500" />
  <item android:drawable="@drawable/rocket2" android:duration="500" />
  <item android:drawable="@drawable/rocket3" android:duration="500" />
 </animation-list>
```

Creating and Using Frame-by-Frame Animations

- Animation-г дэлгэцэнд үзүүлэхдээ setBackgroundResource функцээр тухайн View-н дэвсгэрийг тохируулна.

```
ImageView image = (ImageView)findViewById(R.id.my_animation_frame);
image.setBackgroundResource(R.drawable.animated_rocket);
```

- Animation-г эхлүүлэх

```
AnimationDrawable animation = (AnimationDrawable)image.getBackground();
animation.start();
```

Using Themes to Skin Your Applications

- Theme нь тухайн апп хэрхэн зохицох болон харагдахыг илэрхийлнэ.
- Manifest-ийн application tag дахь android:theme атрибутад Theme-г зааж өгөх бөгөөд эсвэл Activity тус бүрд нь Activity tag дээр нь зааж өгж болно.

Using Themes to Skin Your Applications

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.google.android.home">
  <application
     android:theme="@android:style/Theme.Light" >
     <activity
        android:theme="@android:style/Theme.Black" >
     </activity>
  </application>
</manifest>
```

– Theme.Black – хар дэвсгэртэй, контролууд болон текст нь цагаан

Using Themes to Skin Your Applications

- –Theme.Light – дэвсгэр нь цагаан, контролууд болон текст нь бараан өнгөтэй
- –Theme.Translucent – Хэсэгчлэн тунгалаг загвар

•Програм ажиллах явцад (run-time) activity-н theme тохируулж болох бөгөөд энэ нь зарим тохиолдолд дэмжигддэггүй.

•Тэхдээ үүнийг ашиглах тохиолдолд layout-г дэлгэцэнд дүрслэхээс өмнө дуудах хэрэгтэй.

Using Themes to Skin Your Applications

```
protected void onCreate(Bundle icicle) {
  super.onCreate(icicle);
  setTheme(android.R.style.Theme_Translucent);
  setContentView(R.layout.main);
}
```

Д

theme идэвхжихгүи тохиолдолд анхдагч android.R.style.Theme хэрэгжих болно

Using the Touch Screen

- View эсвэл Activity үүсгээд дэлгэцэнд хүрэх event-үүдийг барих боломжтой.

```
@Override
public boolean onTouchEvent(MotionEvent event) {
  return super.onTouchEvent(event);
}
```

Processing Touch Events

- onTouchEvent нь хэрэглэгч дэлгэцэнд хүрэхэд анх ажиллах бөгөөд байрлал өөрчлөгдөх бүрт 1 удаа дуудагдаж, үйлдэл дуусахад дахин 1 удаа дуудагдана.
- MotionEvent параметрээр event-ийн төрлүүд орж ирэх ба тэдгээрийг getAction функцээр авна.

Using the Touch Screen

```
@Override
public boolean onTouchEvent(MotionEvent event) {

  int action = event.getAction();

  switch (action) {
    case (MotionEvent.ACTION_DOWN)   : // Touch screen pressed
                                       break;
    case (MotionEvent.ACTION_UP)     : // Touch screen touch ended
                                       break;
    case (MotionEvent.ACTION_MOVE)   : // Contact has moved across screen
                                       break;
    case (MotionEvent.ACTION_CANCEL) : // Touch event cancelled
                                       break;

  }

  return super.onTouchEvent(event);
}
```

Using the Touch Screen

- MotionEvent нь бас дэлгэцийн тухайн хүрсэн цэгийн координатыг агуулдаг.
- Тэдгээрийг getX болон getY функцээр авах ба тэдгээр нь тухайн View болон Activity-даа хамааралтай (relative) координат бөгөөд абсолют (absolute) координатыг getRawX, getRawY аргуудаар авна.

```
// Relative screen coordinates.
int xPos = (int)event.getX();
int yPos = (int)event.getY();
```

```
// Absolute screen coordinates.
int xPosRaw = (int)event.getRawX();
int yPosRaw = (int)event.getRawY();
```

Using the Touch Screen

- Мөн MotionEvent нь хэр хүчтэй дарсаныг хэмжих бөгөөд getPressure функцээр авна.
- Ихэвчлэн 0 – дарагдаагүй, 1 – хэвийн дарагдсан гэсэн утгуудыг буцаана
- getSize функцээр touch-ийн талбайн хэмжээг тодорхойлно.
  - 0 маш бага хэмжээгээр
  - 1 хангалттай өргөн хүрээгээр шүргэлцсэн

Tracking Movement

- ACTION_MOVE үед MotionEvent параметр нь туулсан замын мэдээллээ хадгална.
- Энэ нь тухайн Event болон өмнөх onTouchEvent –н хоорондох бүхий л координатуудыг агуулна.
- getHistorySize функцээр хичнээн хөдөлгөөний байрлал байгаа тоог авна.
- getHistorical* функцээр тухайн байрлалын даралт, хэмжээ, байрлал гэх мэт мэдээллийг авна

Tracking Movement

```
int historySize = event.getHistorySize();

for (int i = 0; i < historySize; i++) {
  long time = event.getHistoricalEventTime(i);
  float pressure = event.getHistoricalPressure(i);
  float x = event.getHistoricalX(i);
  float y = event.getHistoricalY(i);
  float size = event.getHistoricalSize(i);

  // TODO: Do something with each point
}
```

Tracking Movement

- Энгийн загвар

```
@Override
public boolean onTouchEvent(MotionEvent event) {

  int action = event.getAction();

  switch (action) {
    case (MotionEvent.ACTION_MOVE)
    {
      int historySize = event.getHistorySize();
```

Tracking Movement

```
      for (int i = 0; i < historySize; i++) {
        float x = event.getHistoricalX(i);
        float y = event.getHistoricalY(i);
        processMovement(x, y);
      }

      float x = event.getX();
      float y = event.getY();
      processMovement(x, y);

      return true;
    }
  }

  return super.onTouchEvent(event);
}

private void processMovement(float _x, float _y) {
  // Todo: Do something on movement.
}
```