院 系 数据科学与计算机学院 学 号 16337341 姓 名 朱志儒

【实验题目】**Android动画实验**

【实验目的】学习Android的逐帧动画、补间动画、属性动画、Shape、菜单、触摸事件等 。

【实验准备】直接从网上或从上传作业的网站上下载并安装Android Studio。

【实验说明】

先通过看视频birdFly.mp4了解本次实验要完成的任务。下面的实验内容把本次实验分成了18个功能来完成，可以只完成其中一部分功能，并在最后指明完成了哪些功能。

【实验内容】

**核心设计**

1、利用逐帧动画（FrameAnimation）实现小鸟动画（ImageView）。

编程说明：RelativeLayout可以加上白云背景。

参考资料：动画.pdf

参考源码：FrameAnim

使用资源：sprite1.png~sprite8.png

2、利用TouchDown事件和TranslateAnimation实现在屏幕上引导小鸟飞行。

参考资料：安卓程序设计（二）p26 或p41（TouchDown可以用于Activity） p14

参考源码：TweenAnimCases

3、利用ImageView.setRotateY()控制鸟飞行的方向。

4、控制鸟飞行的范围不要超出显示范围。

参考资料：动画.pdf可以加一个宽高match\_parent的ImageView(后面13可以用到)，得到它的宽度和高度，或者用屏幕尺寸（好像不准确）。

5、通过取得屏幕大小控制鸟飞行的速度。 如果整个屏幕对角飞行时间为5秒，按飞行距离设置每段飞行的时间，距离越短，时间越短。

6、采用属性动画（500ms）使得鸟飞行方向平滑变化（取代第3条）。注意：飞行动画的启动要延迟500ms以配合本动画。

参考源码：PropAnimCase

**菜单设计**

7、增加选项菜单：开始录像、结束录像、录像重放、结束回放

参考源码：Menu

参考资料：对话框和菜单.pdf

**录像小圆点设计**

8、增加一个表示正在录制的小红点（shape）。设计一个0.5秒显示红点0.5秒显示灰色或透明点的逐帧动画。

参考资料：动画.pdf

参考源码：FrameAnim

9、只有在录像时才显示小红点，结束录像时不显示小红点。

参考资料：列表与布局，帧布局(p43)setVisibility

**录制设计**

10、用一个类Pos和一个ArrayList<Pos>记录所走路径的方向和停留地点。

11、每次开始录像时清除原来的记录。

**重播设计**

12、通过菜单“录像重放”重新显示飞行过程。  
编程说明：（1）开始重放时要清除Path对象。  
 （2）最好把前面的飞行动画用*类方法*独立出来。

（3）动画通过onAnimationEnd()事件接续。

参考资料：安卓程序设计（二） p24

参考源码：PropAnimCase

13、在每条线路飞行前显示飞行路线。

编程说明：（1）增加一个自定义View作为背景（可以用第4步中增加的ImageView），并在其中定义一个Path对象

（2）每次开始一段线路时增加Path对象的点(lineTo)，并调用其方法invalidate()，使其重画Path对象。

（3）可以采用setStrokeJoin和setStrokeCap令路线圆滑。

（4）可以采用BlurMaskFilter令路线圆滑。

参考资料：安卓程序设计（二） p56 p29 p62 p104

参考源码：CanvasDrawTest PathView.java

14、菜单“结束重放”可以中途停止重放，并清除路径的显示。

15、重放完毕后回到播放前的状态。

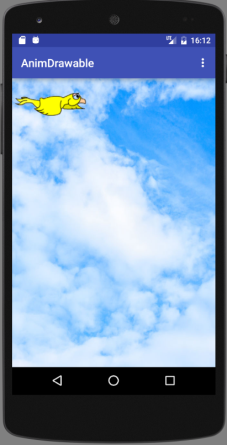
**其他控制功能**

16、播放录像时令触摸操作无效。

17、在飞行进行过程中令触摸操作无效。

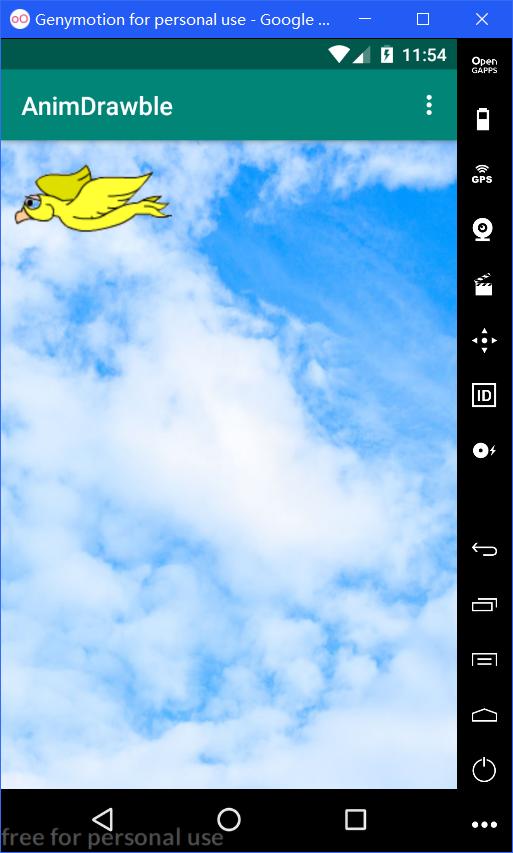
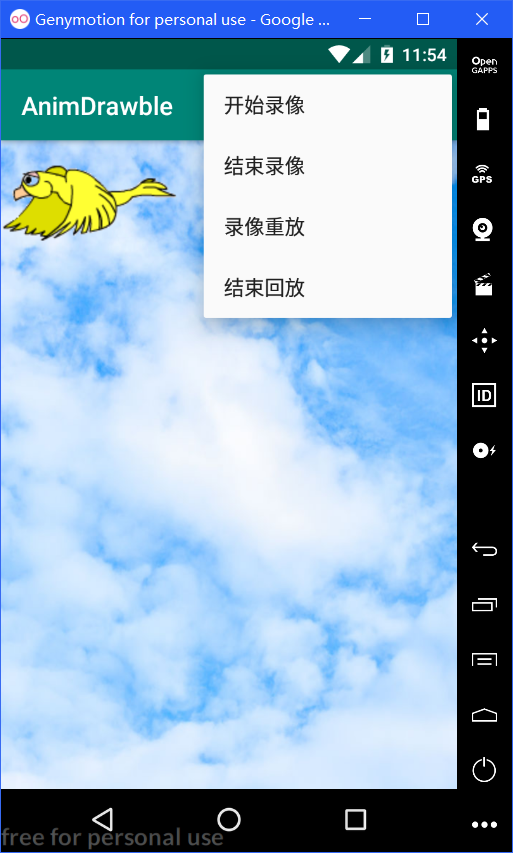
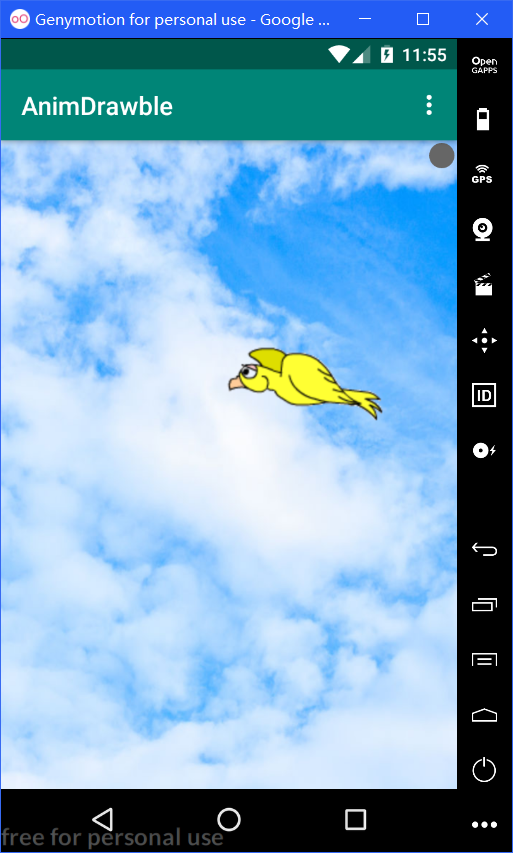
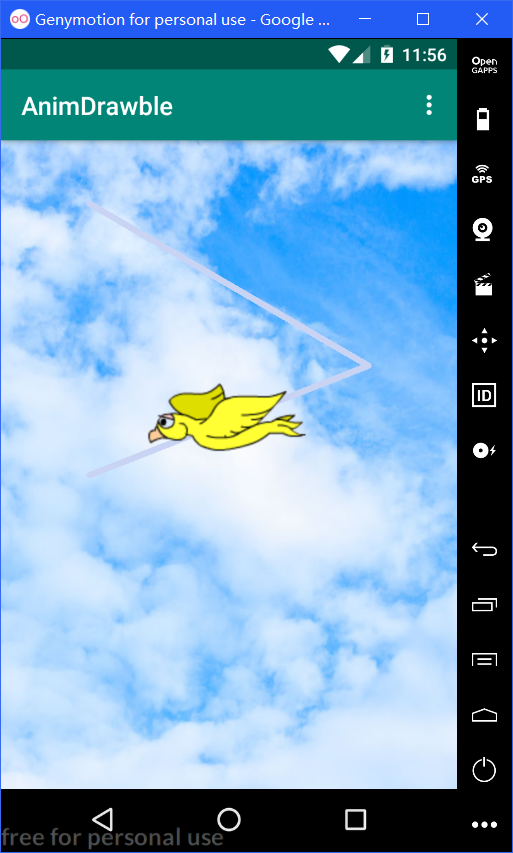
18、录像时不能重放录像。

参考截屏：



*完成后可以录制一个视频，直接用手机或者在教学网站的软件下载中下载Camtasia Studio。*

*按照完成的情况自己选择一些运行界面进行截屏：*

*完成后的主要源码(.java和.xml)：*

MainActivity.java:

public class **MainActivity** extends **AppCompatActivity** implements **View.OnTouchListener**

{

private ImageView imageView, emptyImage, shapeView;

private AnimationDrawable sprite, drawable;

private boolean toRight = true, record = false, play = false, respond = true;

private ArrayList<Pos> pathList = new ArrayList<>();

private Pos lastPos;

private PathView pathView;

@Override

protected void **onCreate**(Bundle savedInstanceState) {

super.**onCreate**(savedInstanceState);

**setContentView**(R.layout.activity\_main);

ConstraintLayout constraintLayout = (ConstraintLayout) **findViewById**(R.id.container);

constraintLayout.**setOnTouchListener**(this);

imageView = (ImageView) **findViewById**(R.id.sprite);

emptyImage = (ImageView) **findViewById**(R.id.empty\_image);

shapeView = (ImageView) **findViewById**(R.id.shape);

pathView = (PathView) **findViewById**(R.id.path);

sprite = (AnimationDrawable) imageView.**getBackground**();

drawable = (AnimationDrawable) shapeView.**getBackground**();

sprite.**start**();

}

@Override

public boolean **onCreateOptionsMenu**(Menu menu) {

**getMenuInflater**().**inflate**(R.menu.main\_menu, menu);

return true;

}

@Override

public boolean **onOptionsItemSelected**(MenuItem item) {

switch (item.**getItemId**()) {

case R.id.start\_record:

record = true;

shapeView.**setVisibility**(View.VISIBLE);

drawable.**start**();

pathList.**clear**();

pathList.**add**(new **Pos**(imageView.**getX**(), imageView.**getY**(), toRight));

break;

case R.id.stop\_record:

record = false;

drawable.**stop**();

shapeView.**setVisibility**(View.GONE);

break;

case R.id.start\_play:

if (record) break;

pathView.**clearPath**();

play = true;

lastPos = new **Pos**(imageView.**getX**(), imageView.**getY**(), toRight);

if (pathList.**size**() != 0) {

Pos firstPos = pathList.**get**(0);

imageView.**setX**(firstPos.**getX**());

imageView.**setY**(firstPos.**getY**());

if (toRight != firstPos.**isToRight**())

if (firstPos.**isToRight**())

imageView.**setRotationY**(0);

else

imageView.**setRotationY**(180);

toRight = firstPos.**isToRight**();

**playRecord**(firstPos, 1);

}

break;

case R.id.stop\_play:

pathView.**clearPath**();

imageView.**clearAnimation**();

imageView.**setX**(lastPos.**getX**());

imageView.**setY**(lastPos.**getY**());

if (toRight != lastPos.**isToRight**())

if (lastPos.**isToRight**())

imageView.**setRotationY**(0);

else

imageView.**setRotationY**(180);

toRight = lastPos.**isToRight**();

play = false;

break;

}

return super.**onOptionsItemSelected**(item);

}

@Override

public boolean **onTouch**(View view, final MotionEvent event) {

if (play || !respond) return true;

switch (event.**getActionMasked**()) {

case MotionEvent.ACTION\_DOWN:

respond = false;

float deX = event.**getX**() + imageView.**getWidth**()/2 > emptyImage.**getWidth**() ? emptyImage.**getWidth**() - imageView.**getWidth**()/2 : event.**getX**();

float deY = event.**getY**() + imageView.**getHeight**()/2 > emptyImage.**getHeight**() ? emptyImage.**getHeight**() - imageView.**getHeight**()/2 : event.**getY**();

final float X = deX < imageView.**getWidth**()/2 ? imageView.**getWidth**()/2 : deX;

final float Y = deY < imageView.**getHeight**()/2 ? imageView.**getHeight**()/2 : deY;

final float toX = X - imageView.**getX**() - imageView.**getWidth**()/2;

final float toY = Y - imageView.**getY**() - imageView.**getHeight**()/2;

if (toX < 0 && toRight) {

toRight = false;

ObjectAnimator rotateY = ObjectAnimator.**ofFloat**(imageView, "rotationY", 0, 180);

rotateY.**setDuration**(500);

rotateY.**start**();

}

else if (toX > 0 && !toRight) {

toRight = true;

ObjectAnimator rotateY = ObjectAnimator.**ofFloat**(imageView, "rotationY", 180, 0);

rotateY.**setDuration**(500);

rotateY.**start**();

}

TranslateAnimation animation = new **TranslateAnimation**(0, toX, 0, toY);

animation.**setStartOffset**(500);

double maxLength = Math.**sqrt**(Math.**pow**(emptyImage.**getHeight**() - imageView.**getHeight**(), 2) + Math.**pow**(emptyImage.**getWidth**() - imageView.**getWidth**(), 2));

double speed = Math.**sqrt**(Math.**pow**(toX, 2) + Math.**pow**(toY, 2)) / maxLength;

animation.**setDuration**(Double.**valueOf**(5000 \* speed).**longValue**());

imageView.**startAnimation**(animation);

animation.**setAnimationListener**(new Animation.**AnimationListener**() {

@Override

public void **onAnimationStart**(Animation animation) {

}

@Override

public void **onAnimationEnd**(Animation animation) {

imageView.**setX**(X - imageView.**getWidth**()/2);

imageView.**setY**(Y - imageView.**getHeight**()/2);

if (record) {

Pos pos = new **Pos**(imageView.**getX**(), imageView.**getY**(), toRight);

pathList.**add**(pos);

}

respond = true;

}

@Override

public void **onAnimationRepeat**(Animation animation) {

}

});

break;

}

return true;

}

private void **playRecord**(Pos firstPos, final int index) {

if (index >= pathList.**size**()) {

return;

}

final Pos nextPos = pathList.**get**(index);

pathView.**setData**(firstPos, nextPos, imageView.**getWidth**(), imageView.**getHeight**());

float toX = nextPos.**getX**() - firstPos.**getX**();

float toY = nextPos.**getY**() - firstPos.**getY**();

if (toX < 0 && toRight) {

toRight = false;

ObjectAnimator rotateY = ObjectAnimator.**ofFloat**(imageView, "rotationY", 0, 180);

rotateY.**setDuration**(500);

rotateY.**start**();

}

else if (toX > 0 && !toRight) {

toRight = true;

ObjectAnimator rotateY = ObjectAnimator.**ofFloat**(imageView, "rotationY", 180, 0);

rotateY.**setDuration**(500);

rotateY.**start**();

}

TranslateAnimation animation = new **TranslateAnimation**(0, toX, 0, toY);

animation.**setStartOffset**(500);

double maxLength = Math.**sqrt**(Math.**pow**(emptyImage.**getHeight**() - imageView.**getHeight**(), 2) + Math.**pow**(emptyImage.**getWidth**() - imageView.**getWidth**(), 2));

double speed = Math.**sqrt**(Math.**pow**(toX, 2) + Math.**pow**(toY, 2)) / maxLength;

animation.**setDuration**(Double.**valueOf**(5000 \* speed).**longValue**());

imageView.**startAnimation**(animation);

animation.**setAnimationListener**(new Animation.**AnimationListener**() {

@Override

public void **onAnimationStart**(Animation animation) {

}

@Override

public void **onAnimationEnd**(Animation animation) {

imageView.**setX**(nextPos.**getX**());

imageView.**setY**(nextPos.**getY**());

**playRecord**(nextPos, index + 1);

}

@Override

public void **onAnimationRepeat**(Animation animation) {

}

});

}

}

PathView.java:

public class **PathView** extends **View** {

private Paint paint = new **Paint**();

private Path path = new **Path**();

private Context context;

public **PathView**(Context context, AttributeSet attributeSet) {

super(context, attributeSet);

this.context = context;

}

@Override

protected void **onDraw**(Canvas canvas) {

super.**onDraw**(canvas);

paint.**setStrokeWidth**(10);

paint.**setStyle**(Paint.Style.FILL);

paint.**setStrokeJoin**(Paint.Join.ROUND);

paint.**setStrokeCap**(Paint.Cap.ROUND);

paint.**setStyle**(Paint.Style.STROKE);

paint.**setColor**(Color.**rgb**(201, 212,244));

canvas.**drawPath**(path, paint);

}

public void **setData**(Pos pos1, Pos pos2, int width, int height) {

path.**moveTo**(pos1.**getX**() + width / 2, pos1.**getY**() + height / 2);

path.**lineTo**(pos2.**getX**() + width / 2, pos2.**getY**() + height / 2);

**invalidate**();

}

public void **clearPath**() {

path.**reset**();

**invalidate**();

}

}

Pos.java:

public class **Pos** {

private float x, y;

private boolean toRight;

public **Pos**(float x, float y, boolean toRight) {

this.x = x;

this.y = y;

this.toRight = toRight;

}

public float **getX**() { return x; }

public float **getY**() { return y; }

public boolean **isToRight**() { return toRight; }

}

shape.xml:

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

<animation-list

*xmlns:android*="http://schemas.android.com/apk/res/android"

*android:oneshot*="false">

<item *android:drawable*="@drawable/ic\_brightness\_1\_red\_24dp" *android:duration*="500"/>

<item *android:duration*="500" *android:drawable*="@drawable/ic\_brightness\_1\_grey\_24dp"/>

</animation-list>

sprite.xml:

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

<animation-list *xmlns:android*="http://schemas.android.com/apk/res/android"

*android:oneshot*="false">

<item *android:drawable*="@drawable/sprite1" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite2" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite3" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite4" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite5" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite6" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite7" *android:duration*="100"/>

<item *android:drawable*="@drawable/sprite8" *android:duration*="100"/>

</animation-list>

activity\_main.xml:

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

<android.support.constraint.ConstraintLayout *xmlns:android*="http://schemas.android.com/apk/res/android"

*xmlns:app*="http://schemas.android.com/apk/res-auto"

*xmlns:tools*="http://schemas.android.com/tools"

*android:layout\_width*="match\_parent"

*android:layout\_height*="match\_parent"

*android:background*="@drawable/cloud"

*android:id*="@+id/container"

*tools:context*=".MainActivity">

<ImageView

*android:layout\_width*="match\_parent"

*android:layout\_height*="match\_parent"

*android:id*="@+id/empty\_image"/>

<com.example.animdrawble.PathView

*android:id*="@+id/path"

*android:layout\_width*="match\_parent"

*android:layout\_height*="match\_parent"/>

<ImageView

*android:layout\_width*="wrap\_content"

*android:layout\_height*="wrap\_content"

*android:id*="@+id/sprite"

*android:background*="@drawable/sprite"

*app:layout\_constraintStart\_toStartOf*="parent"

*app:layout\_constraintTop\_toTopOf*="parent"/>

<ImageView

*android:layout\_width*="wrap\_content"

*android:layout\_height*="wrap\_content"

*android:id*="@+id/shape"

*android:background*="@drawable/shape"

*android:visibility*="gone"

*app:layout\_constraintTop\_toTopOf*="parent"

*app:layout\_constraintEnd\_toEndOf*="parent"/>

</android.support.constraint.ConstraintLayout>

【完成情况】

完成了哪些内容？(√完成 ×未做或未完成)

1 [√] 2 [√] 3[√] 4[√] 5 [√] 6 [√] 7[√] 8[√] 9[√] 10[√]

11 [√] 12 [√] 13[√] 14[√] 15 [√] 16 [√] 17[√] 18[√]

【实验体会】

通过本次实验，我初步掌握了Android的补间动画、逐帧动画、属性动画的绘制方法。