# Threads and Animation

### Aims:

To introduce developing an Android App for interactive and dynamic graphics

### Objectives:

* Extending a SurfaceView class
* Realising a Runnable interface for threading

## 1. Introduction

Here we’ll use Android Studio to develop an interactive and dynamic graphics app.

## 2. Starting a new project

Let’s get started…

From windows start Android Studio and start a new Android Studio project.

Choose an **Empty Activity**, name the project ‘**MyDynamicApp**’ and ensure the language is ‘**Java**’ before you click [Finish]. Wait for Gradle sync and indexing to finish (see bottom right of Android Studio).

From the Project explorer right click on …

**app/java/com.*your-id*. mydynamicapp**

to create a **New / Fragment / Fragment (Blank)**. In the resulting dialog select the ‘Fragment Name:’ to be ‘**FirstFragment**’

Right click on ‘**res**’ directory to create a new Android Resource File, from the resulting dialog set ‘File Name’: to be ‘**nav\_graph**’ and select ‘Resource type :’ to be ‘navigation’. Let gradle add additional libraries when prompted.

In the Design View you will see an empty navigation graph. Hence, choose to add your **fragment\_first** to the graph,

Go to **activity\_main.xml** file, delete the TextView, and hence drag and drop from Containers a **NavHostFragment** onto your design view, hence when prompted, choose the **nav\_graph** you created earlier.

From the nav\_host\_fragment’s attribute inspector, change…

* the id to **nav\_host\_fragment**.
* **layout\_width** and **layout\_height** to **match\_parent**.

## 1. Defining a game object

It will be useful to have a common superclass for objects representing images which we will move around dynamically.

From the Project explorer right click on …

**app/java/com.*your-id*. mydynamicapp**

to add ‘New -> Java Class’ called ‘**MyGameObject**’, ensuring it’s in your app’s package directory. Hence, implement the code…

**public class** MyGameObject {  
 **protected float x**, **y, dx, dy**;  
 **protected** Drawable **image**;  
}

And let Android Studio generate the constructor via Code->Generate to give…

**public class** MyGameObject {  
 **protected float x**, **y**, **dx**, **dy**;  
 **protected** Drawable **image**;  
  
 **public** MyGameObject(**float** x, **float** y, **float** dx, **float** dy, Drawable image) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**dx** = dx;  
 **this**.**dy** = dy;  
 **this**.**image** = image;  
 }  
}

Also add to this class

A field

Paint **p** = **new** Paint();

In the constructor add

**p**.setColor(Color.***RED***);  
**p**.setTextSize(100);

and add to the class a move method, for now something very simple to bounce text around a screen

**void** move(Canvas canvas)  
 {  
 **x**+=**dx**;  
 **y**+=**dy**;  
 **if**(**x**>canvas.getWidth() || **x**<0)  
 **dx**=-**dx**;  
 **if**(**y**>canvas.getHeight() || **y**<0)  
 **dy**=-**dy**;  
 canvas.drawText(**"Hello"**, **x**, **y**, p);  
 }

Save the file.

## 2. Sub classing the SurfaceView class

A problem with sub classing a **View** class is that all drawing happens in the main thread competing with other UI controls. A **SurfaceView** can be drawn on by background threads, ... Views can't. With animation and games you’ll want to use separate threads to keep UI working smoothly.

Here you will sub class **SurfaceView** class for graphics in separate thread and use an instance of **SurfaceHolder** to lock canvas to avoid conflicts.

From the Project explorer right click on …

**app/java/com.*your-id*. mydynamicapp**

to add ‘New -> Java Class’ called ‘**MySurfaceView**’, ensuring it’s in your app’s package directory. Hence, implement the code…

In the resulting MySurfaceView.java file you’ll have a new class defined…

**package** com.example.drawingapp;

**public** **class** MySurfaceView {

}

Subclass **SurfaceView** using the ‘extends’ keyword and implement the **Runnable** interface with the ‘implements’ keyword.

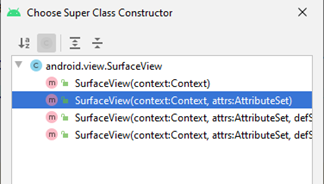
**public class** MySurfaceView **extends** SurfaceView **implements** Runnable{

…

}

Let Android Studio ‘Import View (view.android)’ and

‘Add constructor MySurfaceView(Context, attrs:AttributeSet)’



and add the unimplemented run() method.

So your class definition should now look like…

**package** com.id.dynamicapp;  
**import** android.content.Context;  
**import** android.util.AttributeSet;  
**import** android.view.SurfaceView;  
  
**public class** MySurfaceView **extends** SurfaceView **implements** Runnable{  
 **public** MySurfaceView(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 }  
  
 @Override  
 **public void** run() {  
  
 }  
}

Declare as fields

SurfaceHolder myHolder;

Thread myThread;

**boolean** isRunning=**true**;

Paint pWhite;

Also declare a field for a game object, for example

MyGameObject **myObject**;

In the MySurfaceView constructor method add …

pWhite = **new** Paint();

pWhite.setColor(Color.***WHITE***);

myThread = **new** Thread(**this**);

myThread.start();

myHolder = getHolder();

**myObject** = **new** MyGameObject(100,100,10,10,**null**);

Notably the code declares a new **Thread** object can be associated with an instance of a class providing a **run()** method, by realising **Runnable**.

In this case it the instance of the **MySurfaceView** class, referenced with the ‘this’ keyword.

Also is instantiated a SurfaceHolder object.

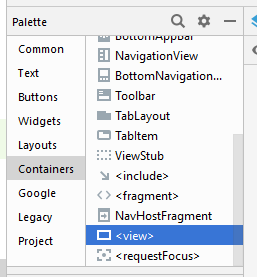
In the run method add…

**while**(**isRunning**)  
{  
 **if**(!**myHolder**.getSurface().isValid())  
 **continue**;  
 Canvas canvas = **myHolder**.lockCanvas();  
 canvas.drawRect(0,0,canvas.getWidth(),canvas.getHeight(), **pWhite**);  
 **myObject**.move(canvas);  
 **myHolder**.unlockCanvasAndPost(canvas);  
}

You can see how the SurfaceHolder lets us test if the canvas is valid and lets us lock and unlock the canvas for thread safe drawing.

Next go to **res/layout/first\_fragment.xml**

From the Designer, right click on the frame layout in the component tree to convert is to a constraint layout, and delete the TextView.

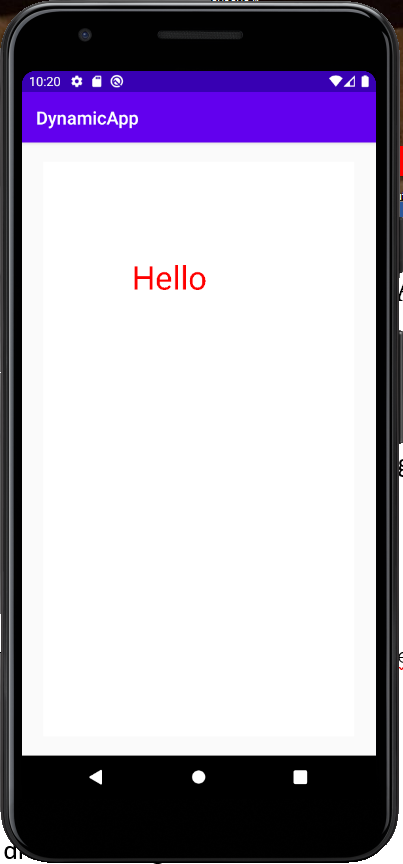


From Containers, drag a <view> onto the design area for this fragment. A dialog box will offer you a choice of classes to associate with this, choose MySurfaceView.

Expand and constrain this view horizontally and vertically, by right clicking on it and choosing Center Horizontally and Center Vertically.

#### 

Save all and ‘Run As’ an Android Application.



## 3. Using Images

Google for a png image of a fish and drag it directly into your project’s drawable directory. Note keep your image’s name simple with no spaces or symbols. In the **MySurfaceView** class’ constructor you could access this resource as a Drawable using

Drawable drawable

= ContextCompat.*getDrawable*(context, R.drawable.*your\_image\_name*);

Where *your\_image\_name* is the id for you fish image.

Then instantiate the game object with, for example…

**myObject** = **new** MyGameObject(100,100,10,10,drawable);

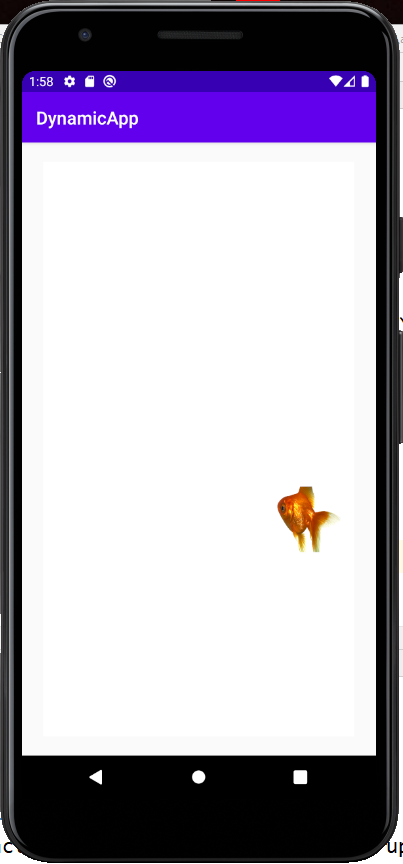
In the **MyGameObject** class define some width and height fields

**float width**=200, **height**=200;

In MyGameObject’s **move()** method you can comment out the drawText, and hence draw the image with…

**image**.setBounds((**int**)**x**,(**int**)**y**,(**int**)(**x**+**width**),(**int**)(**y**+**height**));

**image**.draw(canvas);



Build and run the application.

To avoid the fish going off the view you could easily adjust the if statement in the move method, with for example…

**if**(**x**>(canvas.getWidth()-**width**) || **x**<0)  
 **dx**=-**dx**;  
**if**(**y**>(canvas.getHeight()-**height**) || **y**<0)  
 **dy**=-**dy**;

## EXERCISE

Instead instantiating a single game object, in the **MySurfaceView** class, … declare and instantiate, as a field, a list to hold references to multiple game objects.

ArrayList<MyGameObject> **gameObjs** = **new** ArrayList<MyGameObject>();

You might want to comment out previous game object instantiations.

Then populate the list with references to objects with different start positions, directions and images. For example, if fish images…

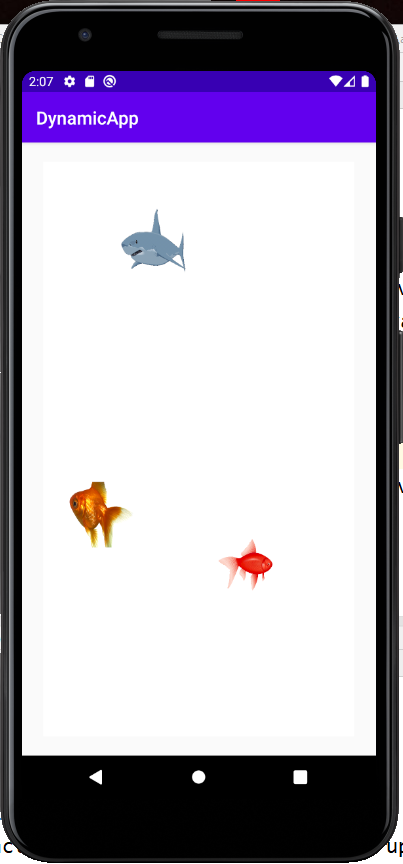
**gameObjs**.add(**new** MyGameObject(100, 500, 2, -2, ContextCompat.*getDrawable*(context, R.drawable.***fish***)));  
  
**gameObjs**.add(**new** MyGameObject(500, 500, -2, 2, ContextCompat.*getDrawable*(context, R.drawable.***fish2***)));  
  
**gameObjs**.add(**new** MyGameObject(300, 300, 4, 4, ContextCompat.*getDrawable*(context, R.drawable.***fish3***)));

In the run method use a ‘for each’ loop, replace myObject.move() with …

**for**(MyGameObject gameObj : **gameObjs**)  
 gameObj.move(canvas);

which will iterate through the game object references and call each of their move methods.

Run the application.



## 4. Managing the thread from your fragment code.

It would be useful to communicate with the **MySurfaceView** object from the rest of code, notably your **FirstFragment** class, which in turn can share information (e.g. settings, high scores, etc) with other fragments via a model.

For example, let’s say we want to turn our thread on an off from the first fragments UI.

Firstly, in the **MySurfaceView** class add public methods to start and stop the thread.

**public** **void** stop()

{

isRunning=**false**;

**while**(**true**)

{

**try** {

myThread.join();

**break**;

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} // block until thread dies

**break**;

}

}

**public** **void** start()

{

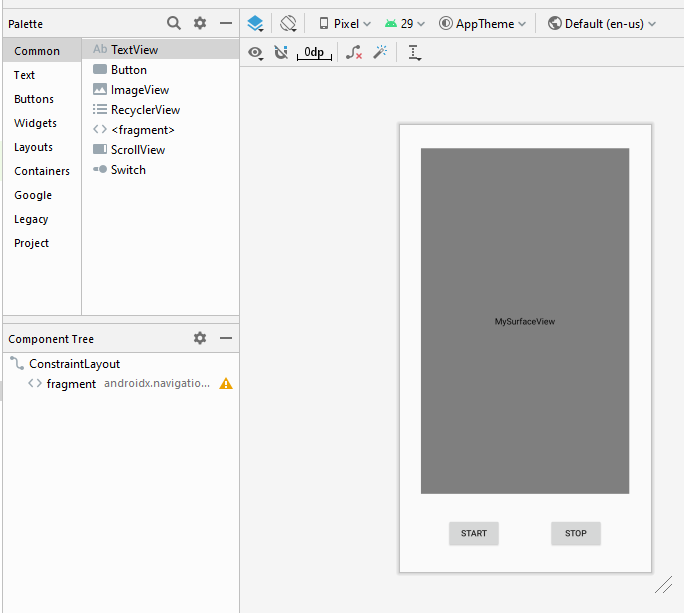
isRunning=**true**;

myThread = **new** Thread(**this**);

myThread.start();

}

In the **fragment\_first.xml** add a couple of buttons [Start] and [Stop], with ids **startButton** and **stopButton**. Taking care to reapply new constraints to your UI.



In the **FIrstFragment** class add as fields some references for the Buttons and the MySurfaceView.

Button **startButton**, **stopButton**;  
MySurfaceView **mySurfaceView**;

Override **onViewCreated()** method to bind the Button and MySurfaceVIew references to their instances.

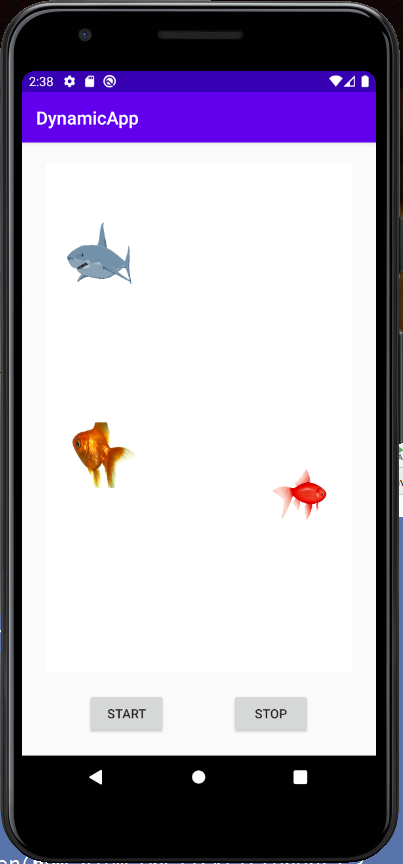
**startButton** = view.findViewById(R.id.***startButton***);  
**stopButton** = view.findViewById(R.id.***stopButton***);  
**mySurfaceView** = view.findViewById(R.id.***view***);

An for each button associate an object to handle clicks events. I.e. associate with you button an instance of anonymous inner classes implementing the **OnClickListener** interface.

**startButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **mySurfaceView**.start();  
 }  
});  
  
**stopButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **mySurfaceView**.stop();  
 }  
});

See how we can simply call **mySurfaceView**.stop() and **mySurfaceView**.start() to control the MySurfaceView instance. Hence, demonstrating how we can use this reference to communicate with public members of MySurfaceView.

Run the application.



**EXERCISE**

If you have time and feel ambitious subclass MyGameObject to create a variety of classes with different move methods hence exploiting the polymorphism of java oop.

Experiment with more a more pleasing fish tank background.

You could create a Player class, and hence instance of player that moves with screen presses. For example, you could subclass MyGameObject to include a method for direction changes. For example…

*// in your Player class***public void** moveToo(**float** X, **float** Y)  
{  
 **if**(X>**x**)  
 **dx**=5;  
 **else  
 dx**=-5;  
 **if**(Y>**y**)  
 **dy**=5;  
 **else  
 dy**=-5;  
}

And use an **OnTouchListener** in **MySurfaceView**, for example…

Player **player**;

OnTouchListener **touchListener** = **new** OnTouchListener() {  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {  
 **player**.moveToo(event.getX(),event.getY());  
 **return true**;  
 }  
};

Taking care to instantiate the player in MySurfaceView’s constructor…

**player**=**new** Player(300, 300, -10, -10, ResourcesCompat.*getDrawable*(getResources(), R.drawable.***diver***, **null**));

**gameObjs**.add(**player**);

And associate the touch listener with the view.

**this**.setOnTouchListener(**touchListener**);

**COMPLETE CODE EXAMPLE**

**MyGameObject.java**

**package** com.id.dynamicapp;  
  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.graphics.drawable.Drawable;  
  
**public class** MyGameObject {  
 **protected float x**, **y**, **dx**, **dy**;  
 **protected** Drawable **image**;  
 **float width**=200, **height**=200;  
 Paint **p** = **new** Paint();  
  
  
 **public** MyGameObject(**float** x, **float** y, **float** dx, **float** dy, Drawable image) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**dx** = dx;  
 **this**.**dy** = dy;  
 **this**.**image** = image;  
 **p**.setColor(Color.***RED***);  
 **p**.setTextSize(100);  
 }  
  
 **void** move(Canvas canvas)  
 {  
 **x**+=**dx**;  
 **y**+=**dy**;  
 **if**(**x**>(canvas.getWidth()-**width**) || **x**<0)  
 **dx**=-**dx**;  
 **if**(**y**>(canvas.getHeight()-**height**) || **y**<0)  
 **dy**=-**dy**;  
 *//canvas.drawText("Hello", x, y, p);* **image**.setBounds((**int**)**x**,(**int**)**y**,(**int**)(**x**+**width**),(**int**)(**y**+**height**));  
 **image**.draw(canvas);  
  
 }  
}

**MySurfaceView.java**

**package** com.id.dynamicapp;  
**import** android.content.Context;  
**import** android.graphics.Canvas;  
**import** android.graphics.Color;  
**import** android.graphics.Paint;  
**import** android.graphics.drawable.Drawable;  
**import** android.util.AttributeSet;  
**import** android.view.SurfaceHolder;  
**import** android.view.SurfaceView;  
  
**import** androidx.core.content.ContextCompat;  
  
**import** java.util.ArrayList;  
  
**public class** MySurfaceView **extends** SurfaceView **implements** Runnable{  
 ArrayList<MyGameObject> **gameObjs** = **new** ArrayList<MyGameObject>();  
 **public** MySurfaceView(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 **pWhite** = **new** Paint();  
 **pWhite**.setColor(Color.***WHITE***);  
 **myHolder** = getHolder();  
 **myThread** = **new** Thread(**this**);  
 **myThread**.start();  
 *//myObject = new MyGameObject(100,100,10,10,null);  
 //Drawable drawable  
 // = ContextCompat.getDrawable(context, R.drawable.fish);  
 //myObject = new MyGameObject(100,100,10,10,drawable);* **gameObjs**.add(**new** MyGameObject(100, 500, 2, -2, ContextCompat.*getDrawable*(context, R.drawable.***fish***)));  
  
 **gameObjs**.add(**new** MyGameObject(500, 500, -2, 2, ContextCompat.*getDrawable*(context, R.drawable.***fish2***)));  
  
 **gameObjs**.add(**new** MyGameObject(300, 300, 4, 4, ContextCompat.*getDrawable*(context, R.drawable.***shark***)));  
  
 }  
  
 SurfaceHolder **myHolder**;  
 Thread **myThread**;  
 **boolean isRunning**=**true**;  
 Paint **pWhite**;  
 MyGameObject **myObject**;  
  
 @Override  
 **public void** run() {  
 **while**(**isRunning**)  
 {  
 **if**(!**myHolder**.getSurface().isValid())  
 **continue**;  
 Canvas canvas = **myHolder**.lockCanvas();  
 canvas.drawRect(0,0,canvas.getWidth(),canvas.getHeight(), **pWhite**);  
 *//myObject.move(canvas);* **for**(MyGameObject gameObj : **gameObjs**)  
 gameObj.move(canvas);  
  
 **myHolder**.unlockCanvasAndPost(canvas);  
 }  
 }  
  
 **public void** start()  
 {  
 **isRunning**=**true**;  
 **myThread** = **new** Thread(**this**);  
 **myThread**.start();  
 }  
  
 **public void** stop()  
 {  
 **isRunning**=**false**;  
 **while**(**true**)  
 {  
 **try** {  
 **myThread**.join();  
 **break**;  
 } **catch** (InterruptedException e) {  
 *//* ***TODO Auto-generated catch block*** e.printStackTrace();  
 } *// block until thread dies* **break**;  
 }  
 }  
}

**FirstFragment.java**

**package** com.id.dynamicapp;  
  
**import** android.os.Bundle;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.annotation.Nullable;  
**import** androidx.fragment.app.Fragment;  
  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.Button;  
  
*/\*\*  
 \* A simple {****@link*** *Fragment} subclass.  
 \* Use the {****@link*** *FirstFragment#newInstance} factory method to  
 \* create an instance of this fragment.  
 \*/***public class** FirstFragment **extends** Fragment {  
  
 *//* ***TODO: Rename parameter arguments, choose names that match*** *// the fragment initialization parameters, e.g. ARG\_ITEM\_NUMBER* **private static final** String ***ARG\_PARAM1*** = **"param1"**;  
 **private static final** String ***ARG\_PARAM2*** = **"param2"**;  
  
 *//* ***TODO: Rename and change types of parameters* private** String **mParam1**;  
 **private** String **mParam2**;  
  
 **public** FirstFragment() {  
 *// Required empty public constructor* }  
  
 */\*\*  
 \* Use this factory method to create a new instance of  
 \* this fragment using the provided parameters.  
 \*  
 \** ***@param param1*** *Parameter 1.  
 \** ***@param param2*** *Parameter 2.  
 \** ***@return*** *A new instance of fragment FirstFragment.  
 \*/  
 //* ***TODO: Rename and change types and number of parameters* public static** FirstFragment newInstance(String param1, String param2) {  
 FirstFragment fragment = **new** FirstFragment();  
 Bundle args = **new** Bundle();  
 args.putString(***ARG\_PARAM1***, param1);  
 args.putString(***ARG\_PARAM2***, param2);  
 fragment.setArguments(args);  
 **return** fragment;  
 }  
  
 @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 **if** (getArguments() != **null**) {  
 **mParam1** = getArguments().getString(***ARG\_PARAM1***);  
 **mParam2** = getArguments().getString(***ARG\_PARAM2***);  
 }  
 }  
  
 @Override  
 **public** View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 *// Inflate the layout for this fragment* **return** inflater.inflate(R.layout.***fragment\_first***, container, **false**);  
 }  
  
 Button **startButton**, **stopButton**;  
 MySurfaceView **mySurfaceView**;  
  
 @Override  
 **public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view, savedInstanceState);  
 **startButton** = view.findViewById(R.id.***startButton***);  
 **stopButton** = view.findViewById(R.id.***stopButton***);  
 **mySurfaceView** = view.findViewById(R.id.***view***);  
 **startButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **mySurfaceView**.start();  
 }  
 });  
  
 **stopButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **mySurfaceView**.stop();  
 }  
 });  
 }  
}

**fragment\_first.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.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:id="@+id/frameLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".FirstFragment"**>  
  
 <**view  
 android:id="@+id/view"  
 class="com.id.dynamicapp.MySurfaceView"  
 android:layout\_width="340dp"  
 android:layout\_height="564dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/startButton"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"** />  
  
 <**Button  
 android:id="@+id/startButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Start"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/stopButton"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/view"** />  
  
 <**Button  
 android:id="@+id/stopButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Stop"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toEndOf="@+id/startButton"  
 app:layout\_constraintTop\_toBottomOf="@+id/view"** />  
</**androidx.constraintlayout.widget.ConstraintLayout**>

**activity\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.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"  
 tools:context=".MainActivity"**>  
  
 <**fragment  
 android:id="@+id/fragment"  
 android:name="androidx.navigation.fragment.NavHostFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:defaultNavHost="true"  
 app:navGraph="@navigation/nav\_graph"** />  
</**androidx.constraintlayout.widget.ConstraintLayout**>