**THE HONG KONG POLYTECHNIC UNIVERSITY**
**Department of Electronic and Information Engineering**

# EIE 3109 Mobile Systems and Application Development
# Assignment

## Introduction:

This assignment is to develop a basic graphics and animation app in Android through the use of SurfaceView, Bitmap, Thread, and Arraylist, and to create simple animations on screen and use TouchEvent for interaction.

**Subject Intended Learning Outcomes** to be assessed in this assignment:

| Part | OC1 | OC2 | OC3 | OC4 |
|---|---|---|---|---|
| I-III | Yes | Yes | Yes | Yes |

Table 1

*Refer to the **Appendix A** for Subject Intended Learning Outcomes and Rubrics*

A. Open a new project with an Empty Activity.

B. **Draw on a SurfaceView**

1. SurfaceView provides a drawing surface which you can control its format, draw objects, and display them the way you want.

Create a SurfaceView and draw an image on the SurfaceView.

**Create Panel.java**

a. Create a Panel class extends **SurfaceView** implements **SurfaceHolder.Callback**

```
public class Panel extends SurfaceView implements SurfaceHolder.Callback
```

b. Add the following override methods. (press **Control+o** to select)

i. **protected void** onDraw(Canvas) (Note: **NOT** draw(Canvas))
ii. **public void** surfaceChanged(SurfaceHolder, **int**, **int**, **int**)
iii. **public void** surfaceCreated(SurfaceHolder)
iv. **public void** surfaceDestroyed(SurfaceHolder)

c. Add a constructor Panel(Context context) and add a SurfaceHolder in the constructor which provides us a canvas we can draw on.

```
public Panel(Context context) {
   super(context);
   getHolder().addCallback(this);
}
```

d. Declare a Bitmap object as a class variable.

```
private Bitmap bmp;
```

e. Initialize the Bitmap object in the constructor (you may copy the launcher icon under mipmap folder to drawable folder or use your own graphic file).

```
bmp = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
```

f. In OnDraw Method, put the image (Bitmap object) on the canvas.

```
protected void onDraw(Canvas canvas) {
   super.onDraw(canvas);
   canvas.drawColor(Color.BLACK);
   canvas.drawBitmap(bmp, 10, 10, null);
}
```

2. Create a Thread class to control the drawing.
**Create GameThread.java**

a. Create GameThread class extends **Thread**.

```
public class GameThread extends Thread
```

b. Declare a SurfaceHolder object and a Panel object for the thread to run and a Boolean variable to control the thread.

```
private SurfaceHolder surfaceHolder;
private Panel panel;
private boolean run = false;
```

c. Add a constructor to receive the SurfaceHolder and Panel objects.

```
public GameThread(SurfaceHolder surfaceHolder, Panel panel) {
   this.surfaceHolder = surfaceHolder;
   this.panel = panel;
}
```

d. Create a set method for run.

```
public void setRunning(boolean run) {
   this.run = run;
}
```

e. Add the override run() method. (Control + o to select)

```
@Override
public void run() {
   super.run();
}
```

3. **In Panel.java**, use the GameThread to draw on Canvas.

a. Declare a GameThread object as a class variable.

```
private GameThread thread;
```

b. Initialize the GameThread object in constructor.

```
thread = new GameThread(getHolder(), this);
```

c. Start the thread when the surface is created.

```
public void surfaceCreated(SurfaceHolder holder) {
   thread.setRunning(true);
   thread.start();
}
```

d. Shut down and wait for thread to finish.

```
public void surfaceDestroyed(SurfaceHolder holder) {
   boolean retry = true;
   thread.setRunning(false);
   while(retry) {
      try {
         thread.join();
         retry = false;
      } catch (InterruptedException e) {
         //keep trying here
      }
   }
}
```

4. **In GameThread.java**

a. Implement the run method. Create a Canvas and lock it. Then draw on the canvas. Unlock and post it to the surfaceHolder.

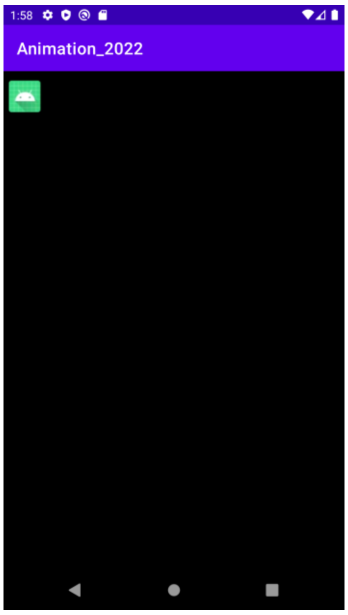
```
public void run() {
   super.run();
   Canvas c;
   while (run) {
      c = null;
      try {
         c = surfaceHolder.lockCanvas();
         synchronized (surfaceHolder) {
            panel.onDraw(c);
         }
      } finally {
          if(c != null) {
             surfaceHolder.unlockCanvasAndPost(c);
          }
      }
   }
}
```

5. **In MainActivity**

a. Set Content View to the Panel Class.

```
setContentView(new Panel(this));
```

6. **Run the program.**



C. **Touch Events**

We will set the Touch Events so that the bitmap moves to where the user clicks on the screen. Touch Events are handled by the Panel class. We need to set the focus to our panel first.

1. **In Panel.java**

a. Set focus in constructor.

```
setFocusable(true);
```

b. Declare 2 int variables as the location of bitmap.

```
private int x=20;
private int y=20;
```

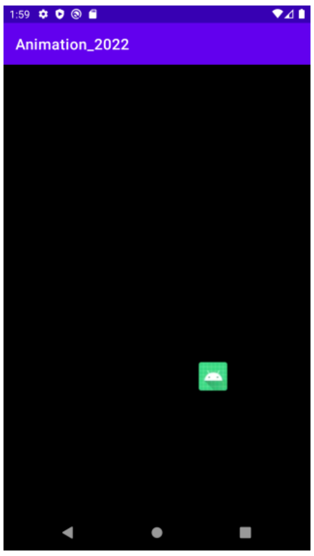
c. Modify the onDraw method to use x and y in drawBitmap method.

```
canvas.drawBitmap(bmp, x, y, null);
```

d. Override the onTouchEvent method to get the new x, y location. (**Control+o** to select)

```
@Override
public boolean onTouchEvent(MotionEvent event) {
   x = (int) event.getX();
   y = (int) event.getY();
   return true;
}
```

2. Run the program and click a few different locations on screen.



3. As you can see, the bitmap did not exactly move to the place you clicked. The place you clicked is actually the upper left corner of the bitmap. Therefore, we replace the new bitmap location to the center by changing the drawBitmap method.

```
canvas.drawBitmap(bmp, x-bmp.getWidth()/2, y-bmp.getHeight()/2, null);
```

4. **Run again.**

D. **Add multiple objects**

1. The Coordinates class is to keep track of the locations of different bitmaps on screen.

   a. **Create Coordinates .java**

```
public class Coordinates {

   private int x = 100;
   private int y = 0;
   private Bitmap bitmap;

   public Coordinates(Bitmap bitmap) {
      this.bitmap = bitmap;
   }

   public void setX(int x) {
      this.x = x - bitmap.getWidth()/2;
   }

   public int getX() {
      return x + bitmap.getWidth()/2;
   }

   public void setY(int y) {
      this.y = y - bitmap.getHeight()/2;
   }

   public int getY() {
      return y + bitmap.getHeight()/2;
   }
}
```

2. **Create GraphicObject class** to hold a bitmap and its coordinates.

   a. Create GraphicObject.java

```
public class GraphicObject {

   private Bitmap bitmap;
   private Coordinates coordinates;

```

```
    public GraphicObject (Bitmap bitmap) {
        this.bitmap = bitmap;
        this.coordinates = new Coordinates(bitmap);
    }

    public Bitmap getGraphic() {
        return bitmap;
    }

    public Coordinates getCoordinates() {
        return coordinates;
    }
}
```

3. **Add GraphicObject to Panel class**. As we don't know how many bitmaps we are adding each time we run the app, we use an ArrayList so that we can add a bitmap anytime we want.

**In Panel.java**

a. Declare an ArrayList of GraphicObject as class variable.
You may **remove x and y variables** in Panel class.

```
private ArrayList<GraphicObject> graphics = new ArrayList<GraphicObject>();
```

b. Modify OnTouchEvent to add new element to the ArrayList when we click on the screen. To add new element only on clicking but not just moving your finger across the screen, add an "if" condition to check if the onTouchEvent is a click event (ACTION_DOWN).

```
public boolean onTouchEvent(MotionEvent event) {
    if(event.getAction() == MotionEvent.ACTION_DOWN) {
        int x = (int) event.getX();
        int y = (int) event.getY();

        GraphicObject graphic = new GraphicObject(bmp);

        int bmpW = graphic.getGraphic().getWidth();
        int bmpH = graphic.getGraphic().getHeight();

        graphic.getCoordinates().setX(x - bmpW/2);
        graphic.getCoordinates().setY(y - bmpH/2);
        graphics.add(graphic);
    }
    return true;
}
```

c. Modify the onDraw method to update all bitmaps in ArrayList.

```
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    Coordinates coords;
    int x, y;
    canvas.drawColor(Color.BLACK);
    for(GraphicObject graphic: graphics) {
        coords = graphic.getCoordinates();
        x = coords.getX();
        y = coords.getY();
        canvas.drawBitmap(bmp, x, y, null);
    }
}
```

4. **Run the program.**

Try to add more bitmaps on screen and your app probably crashes before you can add more than 50 bitmaps on screen. When the onDraw method is called, it updates all the bitmaps in the ArrayList. When you click on the screen while it is updating, the onTouchEvent adds a new bitmap to the ArrayList, hence a **ConcurrentModificationException** is thrown. Therefore, we need to synchronize the thread to prevent this from happening.

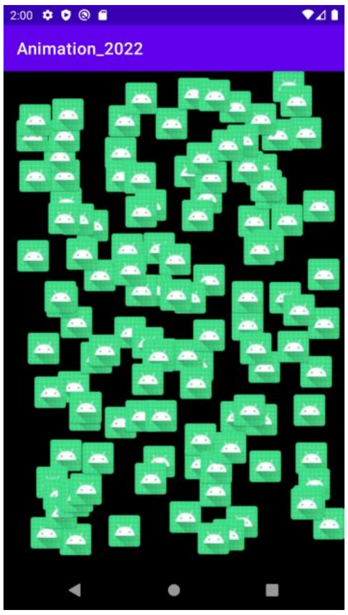
a. **In GameThread.java**
Add a getSurfaceHolder() method.

```
public SurfaceHolder getSurfaceHolder() {
    return surfaceHolder;
}
```

b. **In Panel.java**, modify onTouchEvent method to synchronize the thread.

```
public boolean onTouchEvent(MotionEvent event) {
    synchronized (thread.getSurfaceHolder()) {
        if(event.getAction() == MotionEvent.ACTION_DOWN) {
            int x = (int) event.getX();
            int y = (int) event.getY();
            GraphicObject graphic = new GraphicObject(bmp);
            int bmpW = graphic.getGraphic().getWidth();
            int bmpH = graphic.getGraphic().getHeight();
            graphic.getCoordinates().setX(x - bmpW/2);
            graphic.getCoordinates().setY(y - bmpH/2);
            graphics.add(graphic);
        }
        return true;
    }
}
```

5. **Run the program again.** You should able to add as many bitmaps on the screen as you want.



E. **Moving objects around**

1. The Movement class is to control the direction and speed of the object.

   a. **Create Movement.java**

```java
public class Movement {
   public static final int X_DIRECTION_RIGHT =  1;
   public static final int X_DIRECTION_LEFT  = -1;
   public static final int Y_DIRECTION_DOWN  =  1;
   public static final int Y_DIRECTION_UP    = -1;

   private int xSpeed = 2;
   private int ySpeed = 2;

   private int xDirection = X_DIRECTION_RIGHT;
   private int yDirection = Y_DIRECTION_DOWN;

   public void setXYSpeed(int x, int y) {
      this.xSpeed = x;
      this.ySpeed = y;
   }

   public void setDirections(int xDirection, int yDirection) {
      this.xDirection = xDirection;
```

```
        this.yDirection = yDirection;
    }

    public void toggleXDirection() {
        if(xDirection == X_DIRECTION_RIGHT) {
            xDirection = X_DIRECTION_LEFT;
        } else {
            xDirection = X_DIRECTION_RIGHT;
        }
    }

    public void toggleYDirection() {
        if(yDirection == Y_DIRECTION_UP) {
            yDirection = Y_DIRECTION_DOWN;
        } else {
            yDirection = Y_DIRECTION_UP;
        }
    }

    public int getXSpeed() {
        return xSpeed;
    }

    public int getYSpeed() {
        return ySpeed;
    }

    public int getXDirection() {
        return xDirection;
    }

    public int getYDirection() {
        return yDirection;
    }
}
```

2. **In GraphicObject.java**, add Movement object to GraphicObject class.

   a. Declare Movement object as class variable.

```
private Movement movement;
```

   b. Initialize movement in constructor.

```
this.movement = new Movement();
```

c. Add a get method for movement.

```
public Movement getMovement() {
   return movement;
}
```

3. **In Panel.java**

a. Create an updateMovement method to update the movement of each element in the ArrayList and check if the coordinates are within the screen.

```
public void updateMovement() {
   Coordinates coord;
   Movement movement;

   int x, y;

   for(GraphicObject graphic: graphics) {
      coord = graphic.getCoordinates();
      movement = graphic.getMovement();

      x =
(movement.getXDirection()==Movement.X_DIRECTION_RIGHT)?coord.getX()+mov
ement.getXSpeed():coord.getX()-movement.getXSpeed();
      //check x if reaches border
      if(x < 0) {
         movement.toggleXDirection();
         coord.setX(-x);
      } else if(x + graphic.getGraphic().getWidth() > getWidth()) {
         movement.toggleXDirection();
         coord.setX(x + getWidth() - (x+graphic.getGraphic().getWidth()));
      } else {
         coord.setX(x);
      }

      y =
(movement.getYDirection()==Movement.Y_DIRECTION_DOWN)?coord.getY()+mov
ement.getYSpeed():coord.getY()-movement.getYSpeed();
      //check y if reaches border
      if(y < 0) {
         movement.toggleYDirection();
         coord.setY(-y);
      } else if(y + graphic.getGraphic().getHeight() > getHeight()) {
         movement.toggleYDirection();
         coord.setY(y + getHeight() - (y+graphic.getGraphic().getHeight()));
      } else {
         coord.setY(y);
      }
```

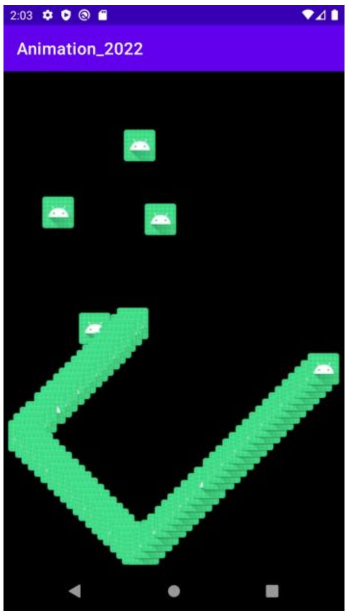
```
    }
}
```

4. **In GameThread.java**

   a. Call the updateMovement method in **run** right before calling the onDraw method.

```
panel.updateMovement();
```

5. **Run the program**. Click the screen to add a few bitmaps. You should see them moving within the screen.



**Your tasks:**

## Part A-E (50%)

### Task 1. Modify the code (10 %)

a. Random speeds are used for x and y directions when adding the bitmap to the screen.

b. Able to remove a bitmap on screen when you click on it.

### Task 2. Design your own animation app (20 %)

You may design an animation game based on Part II or create a brand new one. The game needs to include new features such as:

Suggestions: (No need to do all of them, just suggestions here. You may have your own new features other than those here)

a. Use Sprite for animation.
b. Check for collisions of objects in animation.
c. Use Bluetooth API or WIFI to connect multiple devices.
d. Add audio and video in your app.
e. Use sensors to control the animation.

## Report Writing (20%)

Write a **report for Task 2 (Design your own animation app)** with the following guidelines:

Report body for **Task 2** only:

| | |
|---|---|
| Introduction: | Briefly introduce what your app does. |
| Description: | Classes you defined and APIs you used. |
| Methodology: | Program flows, new features, and how to play your game. |
| Conclusion: | Problems you face and the corresponding solutions. Future developments. |

Limit your report within 10 pages using reasonable font size.
Cite the references (if any) and write what improvements/extensions you add to the source in the report body.
Marks will be deducted if there is a high similarity from other sources (including internet or from your friends) without citation.

**Submission for this assignment:**

1. Upload your android project(s) to Blackboard.
2. Upload a **video demonstration(s)** of your apps to show all tasks you have done.
3. Upload a report to Blackboard.

**Submission item 1 and 2 are mandatory**. You will receive **no marks** for this assignment if you fail to upload the code and video(s) to Blackboard before the 25 November 2022.

## Assessment:

Technical including video demonstration (80%)

Part A – E: 50%
Task 1: 10%
Task 2: 20%

Report (20%)
See "Report Writing" above.

~End~

## Appendix A: Subject Intended Learning Outcomes and Rubrics

**Upon completion of the subject, students will be able to:**

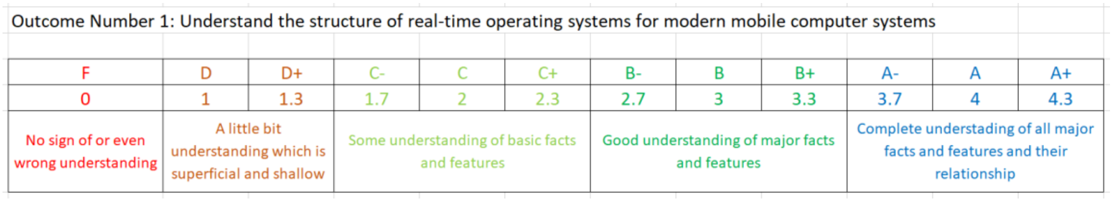
Category A: Professional/academic knowledge and skills

1. Understand the structure of real-time operating systems for modern mobile computer systems.
2. Understand the programming techniques and tools for developing software that is run in modern mobile computer systems
3. Apply the knowledge to develop practical applications for modern real-time mobile computer systems.

Category B: Attributes for all-roundedness

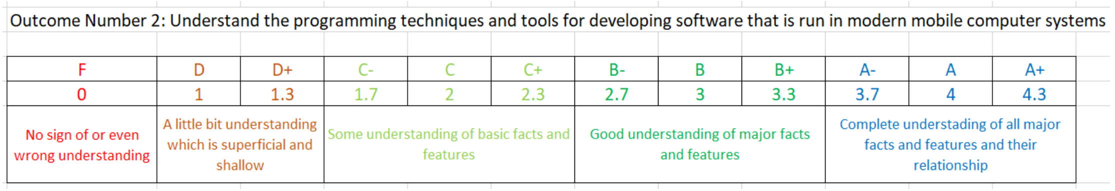
4. Understand the creative process when designing solutions to a problem

### Rubrics to be assessed in this laboratory exercise:



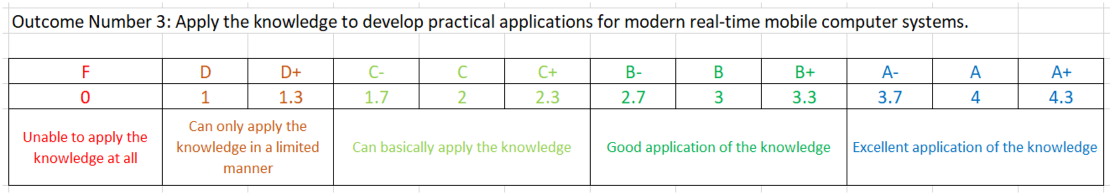
Outcome Number 1: Understand the structure of real-time operating systems for modern mobile computer systems

| F | D | D+ | C- | C | C+ | B- | B | B+ | A- | A | A+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1.3 | 1.7 | 2 | 2.3 | 2.7 | 3 | 3.3 | 3.7 | 4 | 4.3 |
| No sign of or even wrong understanding | A little bit understanding which is superficial and shallow | | Some understanding of basic facts and features | | | Good understanding of major facts and features | | | Complete understading of all major facts and features and their relationship | | |



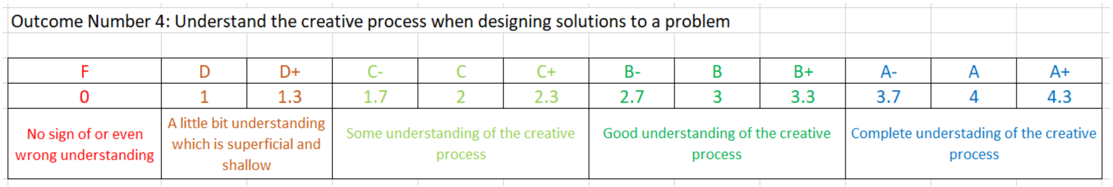
Outcome Number 2: Understand the programming techniques and tools for developing software that is run in modern mobile computer systems

| F | D | D+ | C- | C | C+ | B- | B | B+ | A- | A | A+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1.3 | 1.7 | 2 | 2.3 | 2.7 | 3 | 3.3 | 3.7 | 4 | 4.3 |
| No sign of or even wrong understanding | A little bit understanding which is superficial and shallow | | Some understanding of basic facts and features | | | Good understanding of major facts and features | | | Complete understading of all major facts and features and their relationship | | |



Outcome Number 3: Apply the knowledge to develop practical applications for modern real-time mobile computer systems.

| F | D | D+ | C- | C | C+ | B- | B | B+ | A- | A | A+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1.3 | 1.7 | 2 | 2.3 | 2.7 | 3 | 3.3 | 3.7 | 4 | 4.3 |
| Unable to apply the knowledge at all | Can only apply the knowledge in a limited manner | | Can basically apply the knowledge | | | Good application of the knowledge | | | Excellent application of the knowledge | | |



Outcome Number 4: Understand the creative process when designing solutions to a problem

| F | D | D+ | C- | C | C+ | B- | B | B+ | A- | A | A+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1.3 | 1.7 | 2 | 2.3 | 2.7 | 3 | 3.3 | 3.7 | 4 | 4.3 |
| No sign of or even wrong understanding | A little bit understanding which is superficial and shallow | | Some understanding of the creative process | | | Good understanding of the creative process | | | Complete understading of the creative process | | |