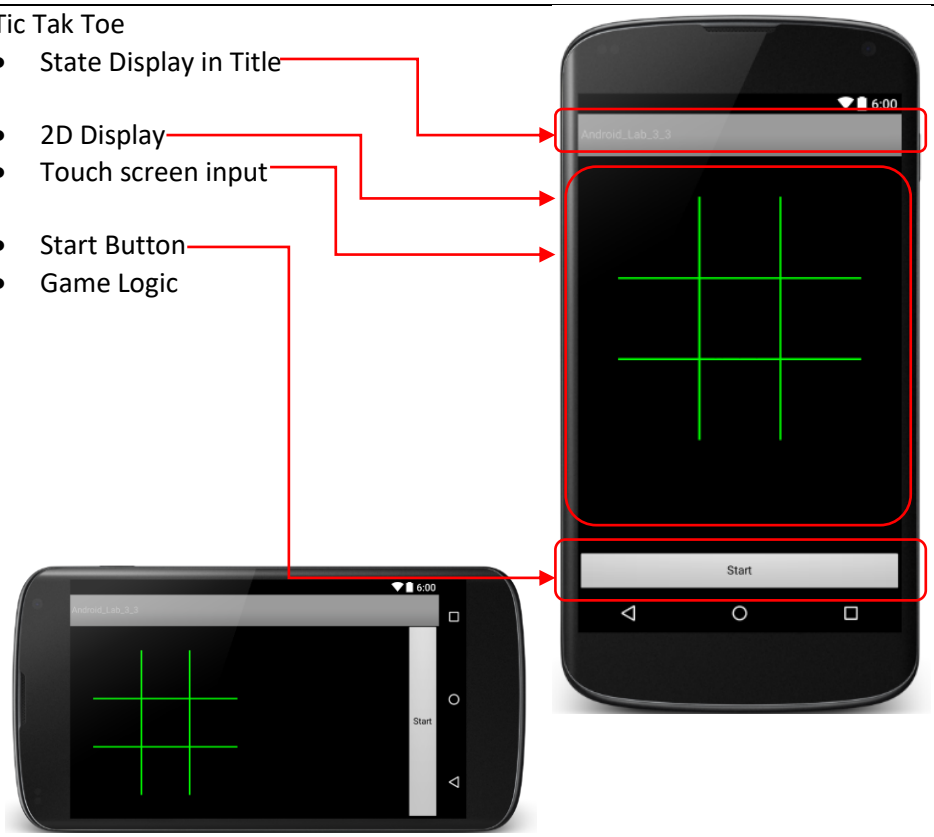
Lab 3	Tic Tak Toe

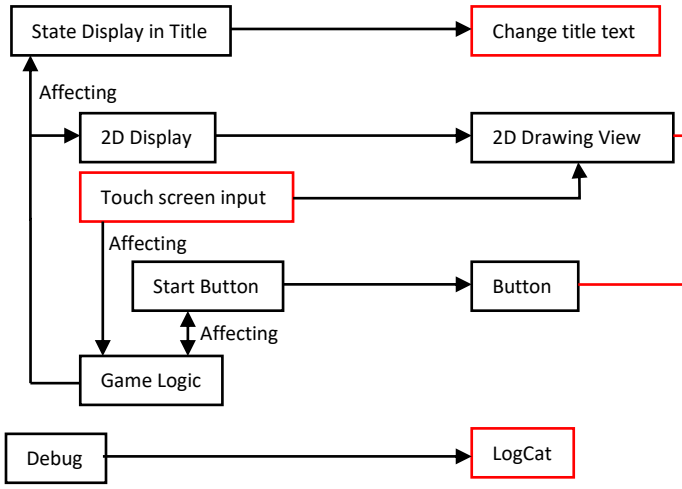
Target

Tic Tak Toe

- State Display in Title
- 2D Display
- Touch screen input
- Start Button
- Game Logic



Program design (To understand what you need and what you have)

You need to have	You learned in last few labs
	TextView (Lab 1) EditText (Lab 1) ImageView (Lab 1) Button (Lab 1) SurfaceView (Lab 2) Accelerometer (Lab 2) Menu (Lab 2)

Program procedure (To plan what should need to do to reach the target)
Task 1 – Set up display
Task 2 – Test touch screen and learn to use LogCat for debugging
Task 3 – Set up display of cross and circle
Task 4 – Write the game logic
Task 1 | Set up display
Knowledge learn in this task:
Layout-land | 1) It is a folder under "res" (resources) 2) It used to store the layout when the apps are in landscape mode. 3) The layout file name in this folder should be same as the file name in "layout" folder
Procedure of the task:
Step 1 Create new project
Project name: Lab_3_1 (or your own one) Build Target: Android 4.4 Package name: ece.course.lab_3_1
Step 2 Create the SurfaceView (Can refer to lab 2)

01) Create a SurfaceView class call "GameView" (Name->GameView, Superclass->SurfaceView)

02) Add the constructor

Code:

```
public GameView(Context context, AttributeSet attrs) {
        super(context, attrs);
}
```

03) Add onDraw Function

04) Add onSizeChanged Function

05) Add variable mDivision (float type)

06) Add "`setWillNotDraw(false);`" inside the constructor

07) Add "`mDivision = ((width < height)? width : height) / 8;`" inside the onSizeChanged

Make the size of grid reference to the size of screen

08) Add function “drawBoard”

Code:

```
private void drawBoard(Canvas canvas) {
        if (canvas == null)
                return;

        canvas.drawColor(Color.BLACK);

        Paint paint = new Paint();
        paint.setColor(Color.GREEN);
        paint.setStrokeWidth(5.0f);

        canvas.drawLine(mDivision * 1, mDivision * 3, mDivision * 7, mDivision * 3, paint);
        canvas.drawLine(mDivision * 1, mDivision * 5, mDivision * 7, mDivision * 5, paint);
        canvas.drawLine(mDivision * 3, mDivision * 1, mDivision * 3, mDivision * 7, paint);
        canvas.drawLine(mDivision * 5, mDivision * 1, mDivision * 5, mDivision * 7, paint);
}
```

Set Background color

Set the width of the lines

Drawing the grid

09) Add “drawBoard(canvas);” inside onDraw

10) Add the SurfaceView and the Button to the file layout -> “activity_main.xml”. (Can refer to lab 1 and lab 2)

Step 3

Make the apps support landscape mode

01) Add a folder “layout-land” under <project> ➔“res”

02) Create and edit a file called “main.xml” in it

03) Like editing the “main.xml” in the “layout”, add the GameView created to the left hand side of the screen and add the button to the right hand side of the phone screen(Refer to page 1, landscape view of phone)

Hints: Change the method in “android:orientation in LinearLayout”,”layout_height & layout_width in button and GameView” of layout-land\main.xml

Step 4

Test the apps by emulator

Task 2	Test touch screen and learn to use logcat for debugging

Knowledge learn in this task:

LogCat	1) Debug tools build in the SDK 2) It can take log and shown in android studio

Procedure of the task:

Step 1

Set up touch screen

01) Open “GameView.java”

02) Add two constant

a. Name: TAG_ON_TOUCH_X Type: String Value: “tagOnTouchX”
b. Name: TAG_ON_TOUCH_Y Type: String Value: “tagOnTouchY”

03) Add variable mHandler(Handler type)

Code:

```
private Handler mHandler;
```

04) Add function onTouchEvent

Code:

```
public boolean onTouchEvent(MotionEvent motionEvent) {
        if (mHandler == null)
                return false;
        int ptrCount = motionEvent.getPointerCount();
        for (int i = 0; i < ptrCount; i++) {
                float tmpX = motionEvent.getX(i);
                float tmpY = motionEvent.getY(i);

                Message msg = mHandler.obtainMessage();
                Bundle bundle = new Bundle();

                bundle.putFloat(TAG_ON_TOUCH_X, tmpX);
                bundle.putFloat(TAG_ON_TOUCH_Y, tmpY);

                msg.setData(bundle);
                mHandler.sendMessage(msg);
        }
        return true;
}
```

This variable contains the information of which points is touched.

Get Back the number of point is touched

Get Back the point is information

05) Add function setHandler

Code:

```
public void setHandler(Handler handler) {
        mHandler = handler;
}
```

Step 2

Update the Activity

01) Open the Activity class

02) Add variables

a. mGameView (GameView Type)
b. btnStart (Button Type)

Code:

```
private GameView mGameView;
private Button btnStart;
```

03) Add following code into onCreate

Code:

```
mGameView = (GameView) findViewById(R.id.mGameView);
mGameView.setHandler(new Handler() {
        public void handleMessage(Message msg) {
                float posX = msg.getData().getFloat(GameView.TAG_ON_TOUCH_X);
                float posY = msg.getData().getFloat(GameView.TAG_ON_TOUCH_Y);
                String tmp = "X: " + posX + ", Y: " + posY;
                setTitle(tmp);          ← Change the title
                Log.i("Msg", tmp);      ← Output to Logcat
                mGameView.invalidate();
        }
});
btnStart = (Button) findViewById(R.id.btnStart);
btnStart.setOnClickListener(new OnClickListener() {
        public void onClick(View view) {
                btnStart.setVisibility(View.INVISIBLE);
                mGameView.invalidate();
        }
});
```

Step 3

Test by emulator

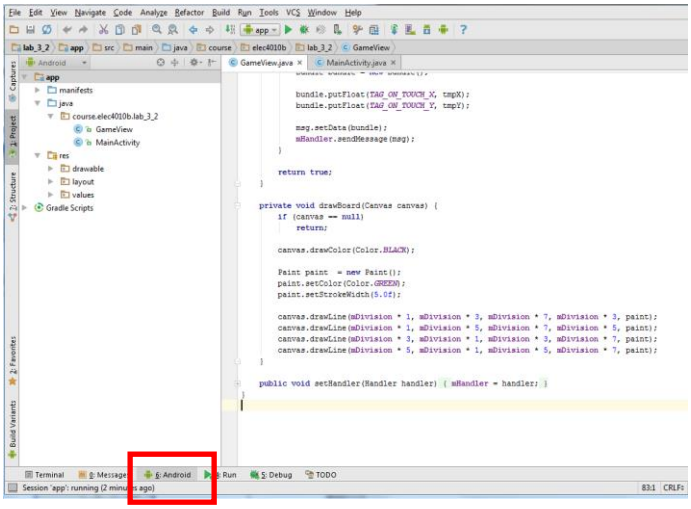
Step 4

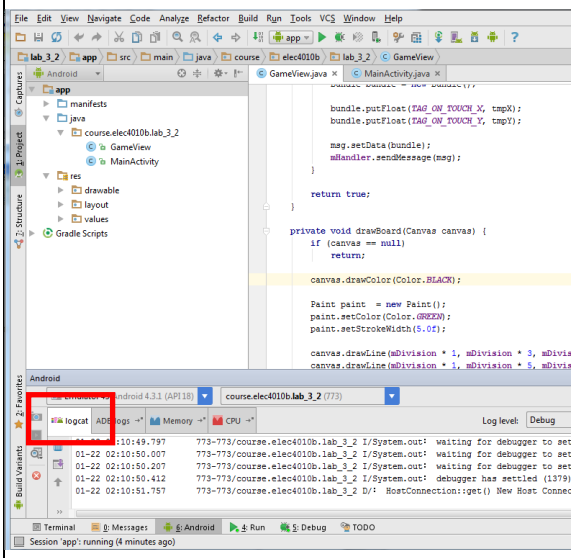
Open LogCat to get back the information

Open LogCat

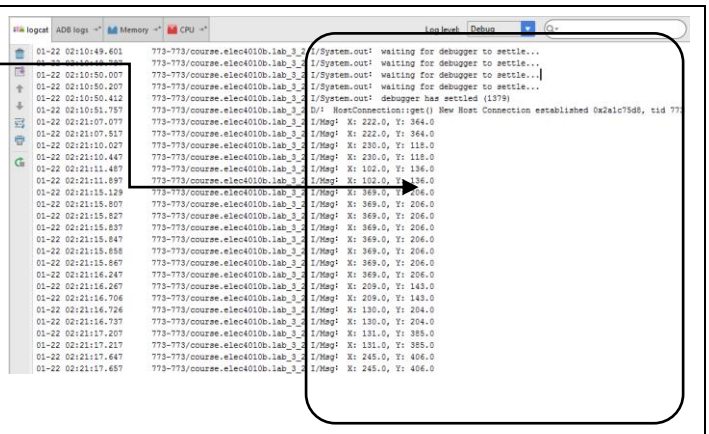
01) Click “Android”

➔ “Select LogCat” View





You can see the LogCat window show up in the right hand side of the eclipse.



Log.i("tag", "msg");

There are 5 fields in it

1. Time: the time received the log
2. 2nd field: type of the log
3. pid: process id, id of the thread.
4. Tag: the tag of the message
5. Message: carried Message

For the Log, there is 5 main type of log, e (Error), w (Warn), I (Info), d (Debug), v (Verbose)
The order of verbosity

Task 3	Set up display of cross and circle

Knowledge learn in this task:

None

Procedure of the task:

Step 1

Update the GameView

01) Add constants

a. Name: LINE_TOP_H Type: int Value: 0
b. Name: LINE_MIDDLE_H Type: int Value: 1
c. Name: LINE_BOTTOM_H Type: int Value: 2
d. Name: LINE_LEFT_V Type: int Value: 3
e. Name: LINE_MIDDLE_V Type: int Value: 4
f. Name: LINE_RIGHT_V Type: int Value: 5
g. Name: LINE_LEFT_D Type: int Value: 6
h. Name: LINE_RIGHT_D Type: int Value: 7
i. Name: PIECE_NONE Type: int Value: 0
j. Name: PIECE_BLUE Type: int Value: 1
k. Name: PIECE_RED Type: int Value: 2

Constant for the lines of three in a row

02) Add variables

a. Name: boardState Type: int[][]
b. Name: hvLines Type: boolean[]

03) Update constructor

Code added:

```
boardState = new int[][] {
        { PIECE_NONE, PIECE_NONE, PIECE_NONE },
        { PIECE_NONE, PIECE_NONE, PIECE_NONE },
        { PIECE_NONE, PIECE_NONE, PIECE_NONE }
};
hvLines = new boolean[] {
        false, false, false, false,
        false, false, false, false
};
```

04) Update onDraw

Code added:

```
for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
                if (boardState[i][j] == PIECE_BLUE) {
                        drawBlueCross(canvas, i, j);
                }
                else if (boardState[i][j] == PIECE_RED) {
                        drawRedCircle(canvas, i, j);
                }
        }
}
if (hvLines[LINE_TOP_H])    drawWinLine(canvas, LINE_TOP_H,    boardState[0][0] == PIECE_BLUE);
if (hvLines[LINE_MIDDLE_H]) drawWinLine(canvas, LINE_MIDDLE_H, boardState[0][1] == PIECE_BLUE);
if (hvLines[LINE_BOTTOM_H]) drawWinLine(canvas, LINE_BOTTOM_H, boardState[0][2] == PIECE_BLUE);
if (hvLines[LINE_LEFT_V])   drawWinLine(canvas, LINE_LEFT_V,   boardState[0][0] == PIECE_BLUE);
if (hvLines[LINE_MIDDLE_V]) drawWinLine(canvas, LINE_MIDDLE_V, boardState[1][0] == PIECE_BLUE);
if (hvLines[LINE_RIGHT_V])  drawWinLine(canvas, LINE_RIGHT_V,  boardState[2][0] == PIECE_BLUE);
if (hvLines[LINE_LEFT_D])   drawWinLine(canvas, LINE_LEFT_D,   boardState[0][0] == PIECE_BLUE);
if (hvLines[LINE_RIGHT_D])  drawWinLine(canvas, LINE_RIGHT_D,  boardState[2][0] == PIECE_BLUE);
```

05) Update onTouchEvent

Code changed to:

```
if (mHandler == null || motionEvent.getAction() != MotionEvent.ACTION_DOWN)
        return false;
int ptrCount = motionEvent.getPointerCount();
for (int i = 0; i < ptrCount; i++) {
        float tmpX = motionEvent.getX(i);
        float tmpY = motionEvent.getY(i);
        if (tmpX > mDivision && tmpX < mDivision * 7 &&
                tmpY > mDivision && tmpY < mDivision * 7) {
                int posX = 0;
                int posY = 0;
                if (tmpX > mDivision * 5) {
                        posX = 2;
                }
                else if (tmpX > mDivision * 3) {
                        posX = 1;
                }
                if (tmpY > mDivision * 5) {
                        posY = 2;
                }
                else if (tmpY > mDivision * 3) {
                        posY = 1;
                }
                Message msg = mHandler.obtainMessage();
                Bundle bundle = new Bundle();
                bundle.putInt(TAG_ON_TOUCH_X, posX);
                bundle.putInt(TAG_ON_TOUCH_Y, posY);
                msg.setData(bundle);
                mHandler.sendMessage(msg);
        }
}
return true;
```

06) Add function drawRedCircle

Code:

```
private void drawRedCircle(Canvas canvas, int posX, int posY) {
        if (canvas == null)
                return;
        Paint paint  = new Paint();
        paint.setColor(Color.RED);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5.0f);
        canvas.drawCircle(mDivision * (posX * 2 + 2), mDivision * (posY * 2 + 2), mDivision - 10, paint);
}
```

Set the draw style

07) Add function drawBlueCross

Code:

```
private void drawBlueCross(Canvas canvas, int posX, int posY) {
	if (canvas == null)
		return;
	Paint paint  = new Paint();
	paint.setColor(Color.BLUE);
	paint.setStrokeWidth(5.0f);
	canvas.drawLine(mDivision * (posX * 2 + 1) + 10, mDivision * (posY * 2 + 1) + 10.0f, mDivision * (posX * 2 + 3) - 10, mDivision *
	(posY * 2 + 3) - 10.0f, paint);

	canvas.drawLine(mDivision * (posX * 2 + 3) - 10, mDivision * (posY * 2 + 1) + 10.0f, mDivision * (posX * 2 + 1) + 10, mDivision *
	(posY * 2 + 3) - 10.0f, paint);
}
```

08) Add function drawWinLine

Code:

```
private void drawWinLine(Canvas canvas, int line, boolean blue) {
	if (canvas == null)
		return;
	Paint paint  = new Paint();
	paint.setColor((blue)? Color.BLUE : Color.RED);
	paint.setStrokeWidth(10.0f);
	switch (line) {
	case LINE_TOP_H :
		canvas.drawLine(mDivision * 2, mDivision * 2, mDivision * 6, mDivision * 2, paint);
		break;
	case LINE_MIDDLE_H :
		canvas.drawLine(mDivision * 2, mDivision * 4, mDivision * 6, mDivision * 4, paint);
		break;
	case LINE_BOTTOM_H :
		canvas.drawLine(mDivision * 2, mDivision * 6, mDivision * 6, mDivision * 6, paint);
		break;
	case LINE_LEFT_V :
		canvas.drawLine(mDivision * 2, mDivision * 2, mDivision * 2, mDivision * 6, paint);
		break;
	case LINE_MIDDLE_V :
		canvas.drawLine(mDivision * 4, mDivision * 2, mDivision * 4, mDivision * 6, paint);
		break;
	case LINE_RIGHT_V :
		canvas.drawLine(mDivision * 6, mDivision * 2, mDivision * 6, mDivision * 6, paint);
		break;
	case LINE_LEFT_D :
		canvas.drawLine(mDivision * 2, mDivision * 2, mDivision * 6, mDivision * 6, paint);
		break;
	case LINE_RIGHT_D :
		canvas.drawLine(mDivision * 2, mDivision * 6, mDivision * 6, mDivision * 2, paint);
		break;
	}
}
```

09) Add function cleanAll

Code:

```
public void cleanAll() {
        for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 3; j++) {
                        boardState[i][j] = PIECE_NONE;
                }
        }
        for (int i = 0; i < 8; i++) {
                hvLines[i] = false;
        }
}
```

10) Add function setBlueCross

Code:

```
public void setBlueCross(int posX, int posY) {
        boardState[posX][posY] = PIECE_BLUE;
}
```

11) Add function setRedCircle

Code:

```
public void setRedCircle(int posX, int posY) {
        boardState[posX][posY] = PIECE_RED;
}
```

12) Add function setWinLine

Code:

```
public void setWinLine(int line) {
        if (line < 0 || line >= 8)
                return;
        hvLines[line] = true;
}
```

Step 2

Update the strings.xml

Add followings string into the file: (refer to lab 1)

1) Name: turn_red Value: “Red Circle Turn...”
2) Name: turn_blue Value: “Blue Cross Trun...”
3) Name: win_blue Value: “Blue Cross Win!!”
4) Name: win_red Value: “Red Circle Win!!”
5) Name: wait_start Value: “Press \"Start\" To Start A Game”
6) Name: draw_game Value: “Draw!!”

Step 3

Update the Activity

01) Add constants

01) Name: PIECE_NONE Type: int Value: 0
02) Name: PIECE_BLUE Type: int Value: 1
03) Name: PIECE_RED Type: int Value: 2

04) Name: STATE_NOT_START Type: int Value: 0
05) Name: STATE_PLAYING Type: int Value: 1
06) Name: STATE_BLUE_WIN Type: int Value: 2
07) Name: STATE_RED_WIN Type: int Value: 3
08) Name: STATE_DRAW_GAME Type: int Value: 4

09) Name: TAG_GAME_STATE Type: String Value: "tagGameState"
10) Name: TAG_IS_BLUE_TURN Type: String Value: "tagIsBlueTurn"
11) Name: TAG_LINE_LEFT Type: String Value: "tagLineLeft"
12) Name: TAG_LINE_MIDDLE Type: String Value: "tagLineMiddle"
13) Name: TAG_LINE_RIGHT Type: String Value: "tagLineRight"
14) Name: TAG_WIN_LINE Type: String Value: "tagWinLine"

02) Add variables

01) Name: boardState Type: int[][] Value: new int[3][3]
02) Name: hvWinLine Type: boolean[] Value: new boolean[8]
03) Name: isBlueTurn Type: boolean Value: true
04) Name: gameState Type: int Value: STATE_NOT_START

03) Update onCreate

01) Add "setTitle(R.string.wait_start);" after "setContentView(R.layout.main);"

02) Update function **handleMessage** inside the **mGameView.setHandler**

Code changed to:

```
if (gameState != STATE_PLAYING)
	return;
int posX = msg.getData().getInt(GameView.TAG_ON_TOUCH_X);
int posY = msg.getData().getInt(GameView.TAG_ON_TOUCH_Y);
inputPiece(posX, posY);
mGameView.invalidate();
```

03) Update function **onClick** inside **btnStart.setOnClickListener**

Code changed to:

```
gameState = STATE_PLAYING;
if (isBlueTurn)
	setTitle(R.string.turn_blue);
else
	setTitle(R.string.turn_red);
btnStart.setVisibility(View.INVISIBLE);
for (int i = 0; i < 3; i++) {
	for (int j = 0; j < 3; j++) {
		boardState[i][j] = PIECE_NONE;
	}
}
for (int i = 0; i < 8; i++)
	hvWinLine[i] = false;
mGameView.cleanAll();
mGameView.invalidate();
```

04) Add function onSaveInstanceState

Code:

```
public void onSaveInstanceState(Bundle outState) {
	outState.putInt(TAG_GAME_STATE,	gameState);
	outState.putBoolean(TAG_IS_BLUE_TURN,  isBlueTurn);
	outState.putIntArray(TAG_LINE_LEFT,	boardState[0]);
	outState.putIntArray(TAG_LINE_MIDDLE,  boardState[1]);
	outState.putIntArray(TAG_LINE_RIGHT,	boardState[2]);
	outState.putBooleanArray(TAG_WIN_LINE, hvWinLine);
}
```

Record the state.
Since if change of the orientation, the state will be clean up.
The function will be called when the orientation change.

05) Add function onResumeInstanceState (Getting back the recorded state)

Code:

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
	gameState = savedInstanceState.getInt(TAG_GAME_STATE, STATE_NOT_START);
	isBlueTurn = savedInstanceState.getBoolean(TAG_IS_BLUE_TURN, true);
	boardState[0] = savedInstanceState.getIntArray(TAG_LINE_LEFT);
	boardState[1] = savedInstanceState.getIntArray(TAG_LINE_MIDDLE);
	boardState[2] = savedInstanceState.getIntArray(TAG_LINE_RIGHT);
	hvWinLine = savedInstanceState.getBooleanArray(TAG_WIN_LINE);
	if (gameState == STATE_PLAYING) {
		btnStart.setVisibility(View.INVISIBLE);
		if (isBlueTurn)
			setTitle(R.string.turn_blue);
		else
			setTitle(R.string.turn_red);
	}
	else {
		btnStart.setVisibility(View.VISIBLE);
		switch (gameState) {
			case STATE_NOT_START : setTitle(R.string.wait_start); break;
			case STATE_BLUE_WIN :  setTitle(R.string.win_blue);   break;
			case STATE_RED_WIN :   setTitle(R.string.win_red);    break;
			case STATE_DRAW_GAME : setTitle(R.string.draw_game);  break;
		}
	}
	mGameView.cleanAll();
	for (int i = 0; i < 3; i++) {
		for (int j = 0; j < 3; j++) {
		if (boardState[i][j] == PIECE_BLUE)
			mGameView.setBlueCross(i, j);
		else if (boardState[i][j] == PIECE_RED)
			mGameView.setRedCircle(i, j);
		}
	}
	for (int i = 0; i < 8; i++)
		if (hvWinLine[i])
			mGameView.setWinLine(i);
	mGameView.invalidate();
}
```

06) Add function inputPiece

Code:

```
private void inputPiece(int posX, int posY) {
	if (boardState[posX][posY] != PIECE_NONE)
		return;
	if (isBlueTurn) {
		boardState[posX][posY] = PIECE_BLUE;
		mGameView.setBlueCross(posX, posY);
		isBlueTurn = false;
		setTitle(R.string.turn_red);
	}
	else {
		boardState[posX][posY] = PIECE_RED;
		mGameView.setRedCircle(posX, posY);
		isBlueTurn = true;
		setTitle(R.string.turn_blue);
	}
}
```

Step 4

Test the code by emulator

Task 4	Write the game logic

Knowledge learn in this task:

None

Procedure of the task:

Step 1

Update the inputPiece in the Activity

The logic need to implement by yourself.
Followings is the guidelines:
Only need to change the code in <u>inputPiece</u>

Flows

01) Check input is a empty cell or not
02) Check which color turn and apply the corresponding sign into cell
03) Check the game is won by one of the player or not, and do the corresponding action for win.
04) Change the turn to opposite player
05) Check is it draw game

Useful Variables

01) posX function input
02) posY function input

03) boardState class variable
04) isBlueTurnclass variable
05) mGameView class variable
06) hvWinLine class variable
07) gameState class variable
08) btnStart class variable

Useful Constants

01)	PIECE_NONE	From: activity	Type: int
02)	PIECE_BLUE	From: activity	Type: int
03)	PIECE_RED	From: activity	Type: int
04)	STATE_NOT_START	From: activity	Type: int
05)	STATE_PLAYING	From: activity	Type: int
06)	STATE_BLUE_WIN	From: activity	Type: int
07)	STATE_RED_WIN	From: activity	Type: int
08)	STATE_DRAW_GAME	From: activity	Type: int
09)	LINE_TOP_H	From: GameView	Type: int
10)	LINE_MIDDLE_H	From: GameView	Type: int
11)	LINE_BOTTOM_H	From: GameView	Type: int
12)	LINE_LEFT_V	From: GameView	Type: int
13)	LINE_MIDDLE_V	From: GameView	Type: int
14)	LINE_RIGHT_V	From: GameView	Type: int
15)	LINE_LEFT_D	From: GameView	Type: int
16)	LINE_RIGHT_D	From: GameView	Type: int
17)	GONE	From: View	Type: int
18)	VISIBLE	From: View	Type: int
19)	INVISIBLE	From: View	Type: int

Useful Functions

01)	setBlueCross	From: GameView	Variable: posX (int), posY (int)
02)	setRedCircle	From: GameView	Variable: posX (int), posY (int)
03)	setWinLine	From: GameView	Variable: line (int)
04)	setTitle	From: activity	Variable: titleId (int)
05)	setVisibility	From: View	Variable: visibility (int)

Useful Resources

01) R.string.win_blue
02) R.string.win_red
03) R.string.turn_blue
04) R.string.turn_red
05) R.string.draw_game

Step 2

Test in the emulator

Step 3

Export as an apps (refer to lab 2) and demonstrates to the TA / IA

Demo Details:

Case1 Red WIN, Case2 Blue WIN, Case3 Draw Game