Exp. 2: Chess

2-1 Discussion

Chess, also called European chess or International chess, is a two-player strategy board game played on a chessboard, which is estimated to have 10^{43} to 10^{50} changes. A chessboard has 64 squares arranged in an eight-by-eight grid. Black and white pieces are respectively 16, normally made of woods, plastics or stones.

A chessboard consists of 64 squares (eight rows and eight columns). The squares are arranged in two alternating colors (light and dark). When placing the chessboard, the light color square must be at the bottom right corner of two players.

There are two colors when it comes to chess game. One player is in control of the white (or light) pieces, and the other player is in control of the black (or dark) pieces. The player controlling the white pieces is often simply referred to as "white" and the player controlling the black pieces is often simply referred to as "black". Each player has 16 pieces: 1 king, 1 queen, 2 rooks, 2 bishops, 2 knights, and 8 pawns. The chess pieces are then arranged in starting position square before the game starts.

For more the related information, please query at the following URL, http://en.wikipedia.org/wiki/Chess.

2-2 Objective

When planning, compiling Android APP, sometimes we might edit or revise program on different computers, or to use external data, what we need to know is how to import file to project. In this chapter, we are going to teach students how to import file to project through importing files. Moreover, in the process of developing, we will not always execute program on the same computer; hence, we will teach students how to convert project to zip file, which is more convenient for developing.

2-3 Procedure

1. Please download SVN at the following URL, http://tortoisesvn.net/downloads.html; follow the operating system to download corresponded insatallation program, as shown in Figure 2-1. Or to execute installation file in "Software" >> "SVN" file in installation disc. SVN is a version control system of open source code. For the further information, please refer to https://en.wikipedia.org/wiki/Apache_Subversion.

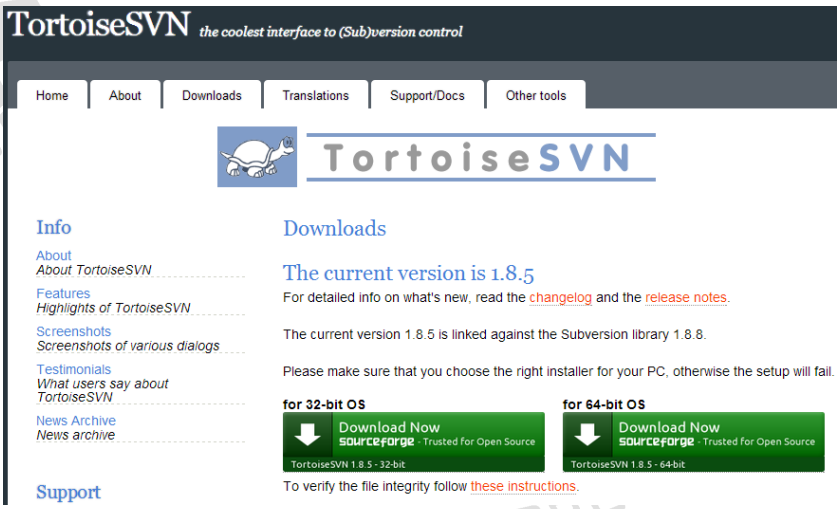

Figure 2-1

2. After downloading SVN, click Install and follow the installation steps, as shown in Figure 2-2 to 2-7.

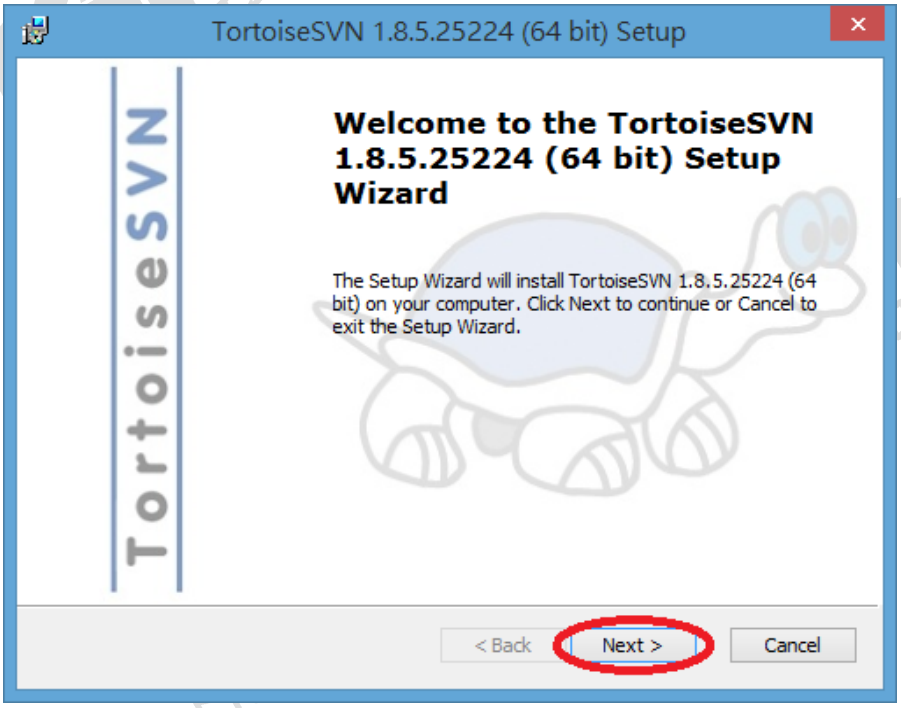

Figure 2-2

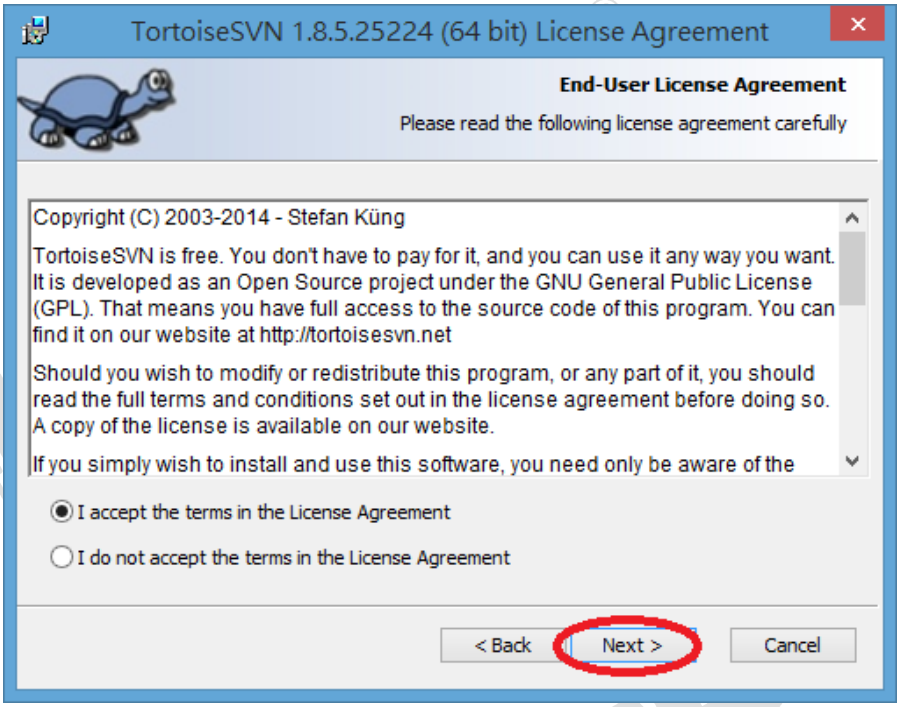

Figure 2-3

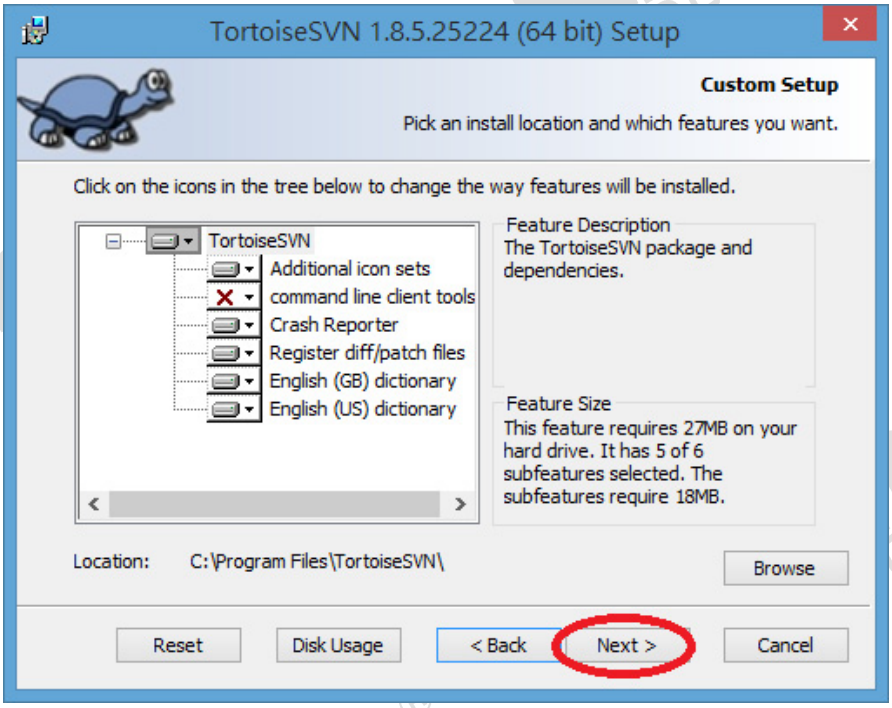

Figure 2-4

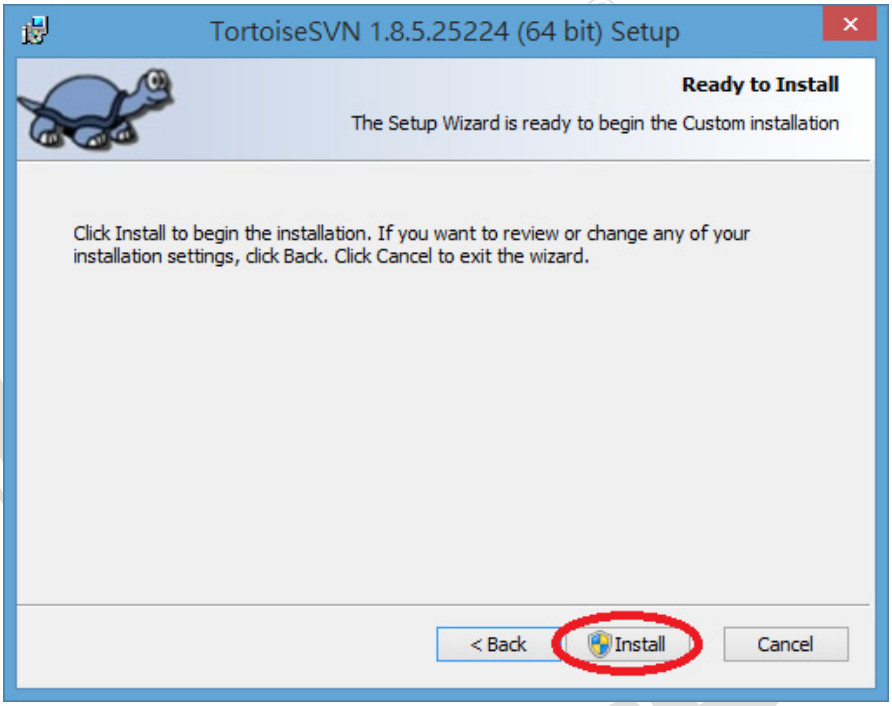

Figure 2-5

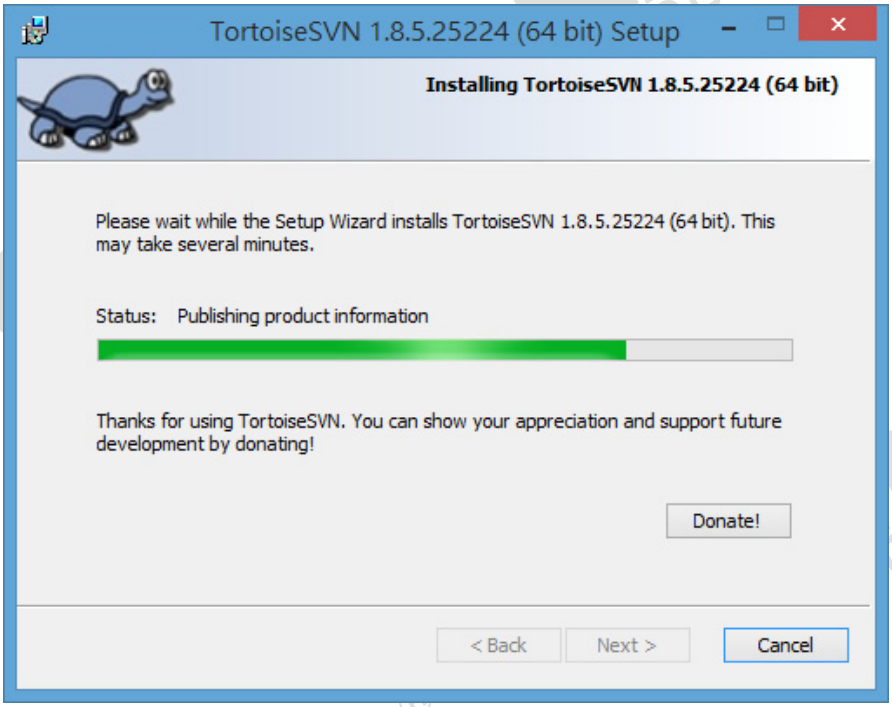

Figure 2-6

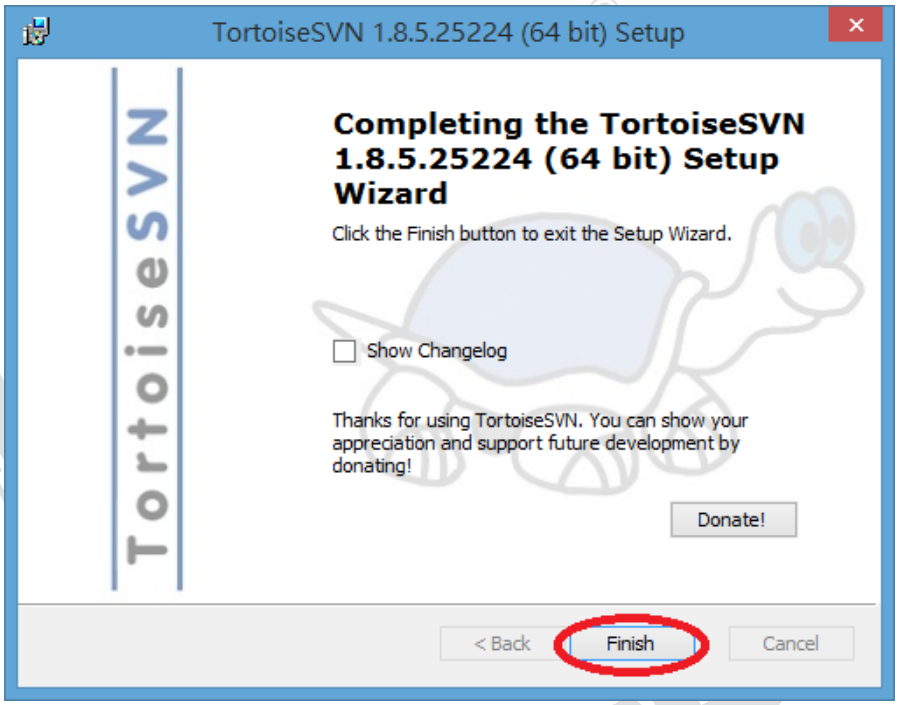

Figure 2-7

3. After the installation, we are going to download the source code of Chess at the following URL, https://code.google.com/p/pocket-chess-for-android/source/checkout. Here we need to copy the part in blue in Figure 2-8.

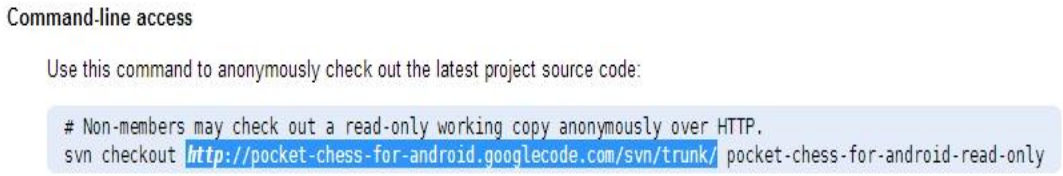

Figure 2-8

4. Please create a directory; there is no restriction on naming. For example, create a directory on desktop and name it as Chess, then click right on directory and select "SVN Checkout..." as shown in Figure 2-9.

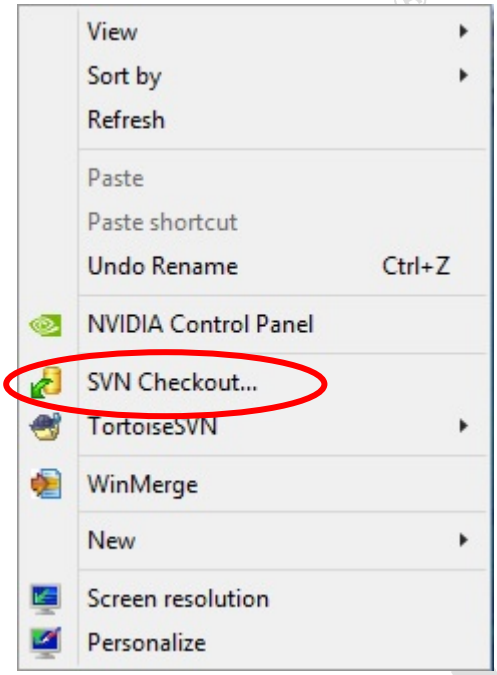

Figure 2-9

5. After clicking, a dialog box as shown in Figure 2-10 will pop out; please note the part in blue on the path, and then compare to the URL on manual. http://pocket-chess-for-android.googlecode.com/svn.

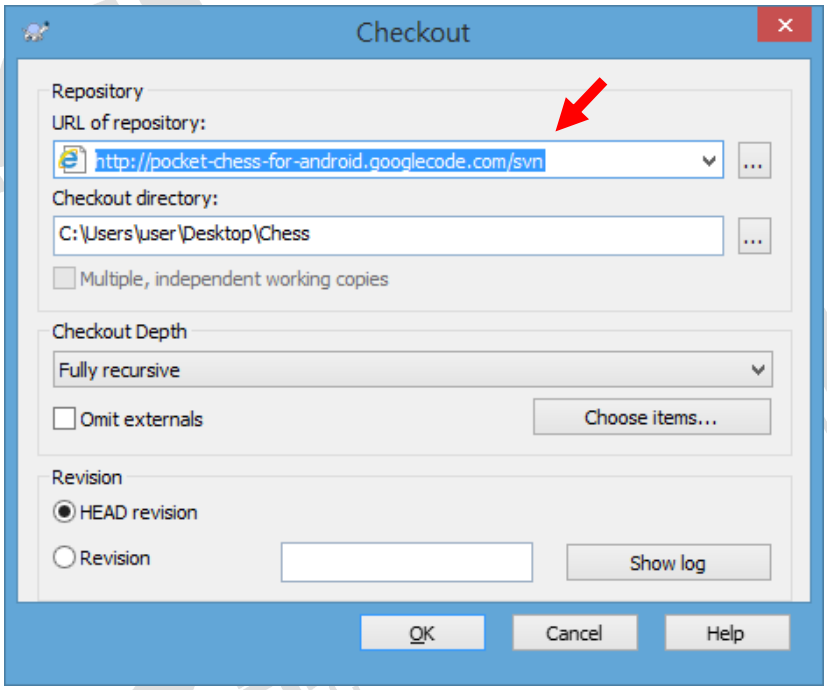

Figure 2-10

6. After the setting is done, click "OK" to start download source code, as shown in Figure 2-11.

Figure 2-11

7. Open Eclipse; click File >> Import to import the source code which is saved in Chess file to Eclipse, as shown in Figure 2-12.

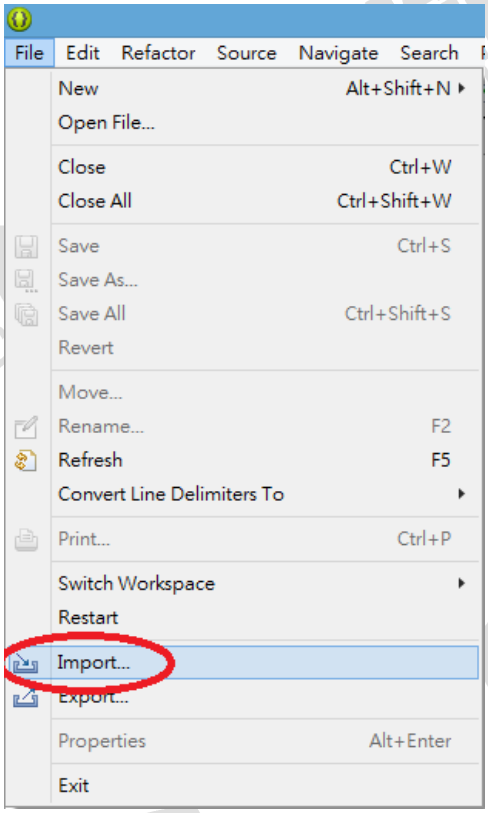

Figure 2-12

8. When the "Import" dialog box is shown, here we need to select General >> Existing Projects into Workspace, as shown in Figure 2-13, and then click "Next" to do the next step.

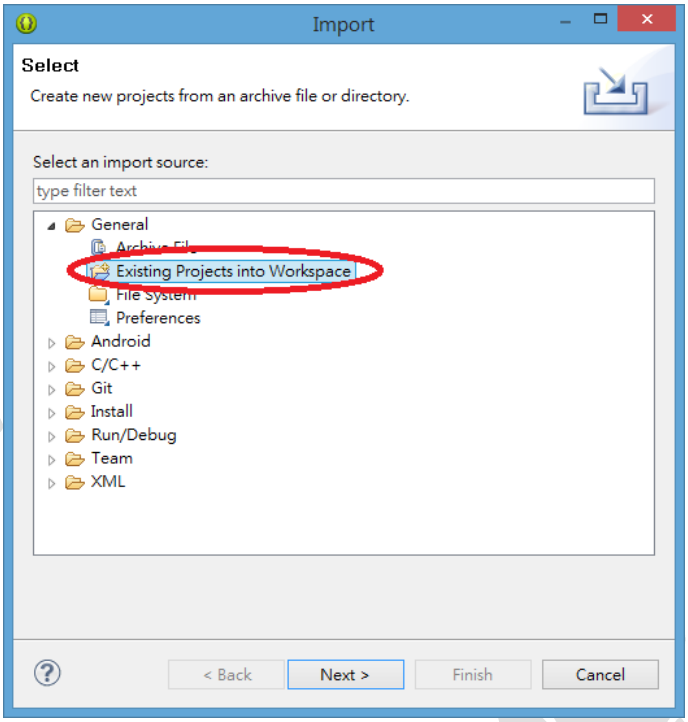

Figure 2-13

9. Click "Browse" in Select root directory, file selecting window will pop out. To the previously created file, Source Code directory, select the directory of "New Chess", as shown in Figure 2-14, and then click "OK".

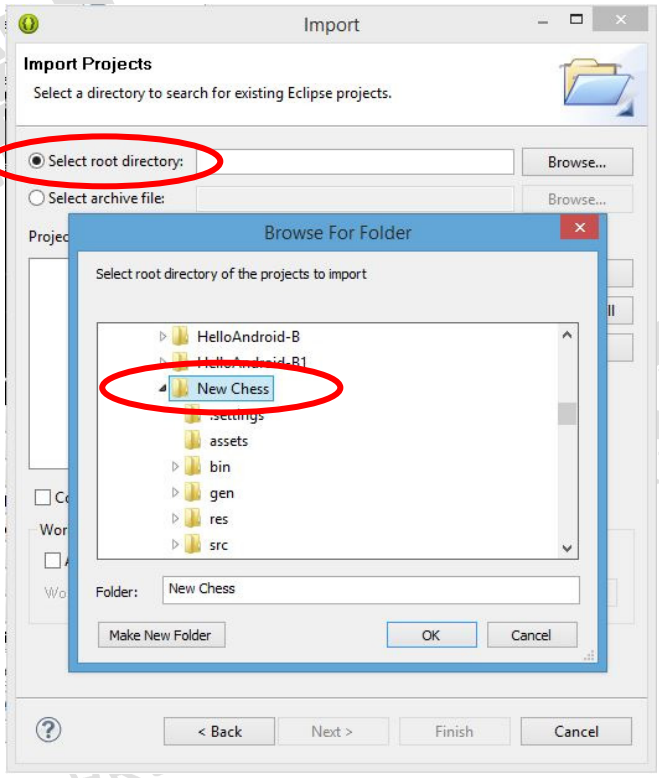

Figure 2-14

10. Click "OK" it will back to the previous dialog box and automatically show the the name of "Projects", as shown in Figure 2-15.

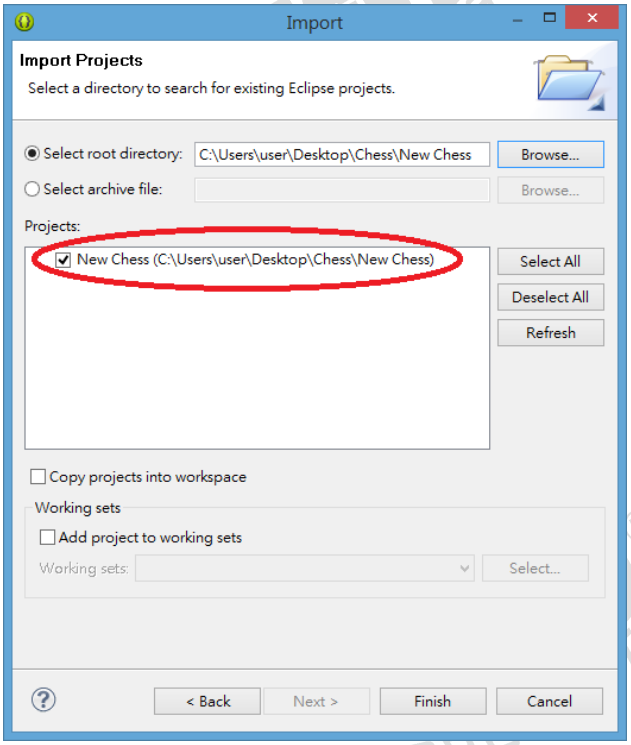

Figure 2-15

11. Click "Finish" to finish importing, as shown in Figure 2-16.

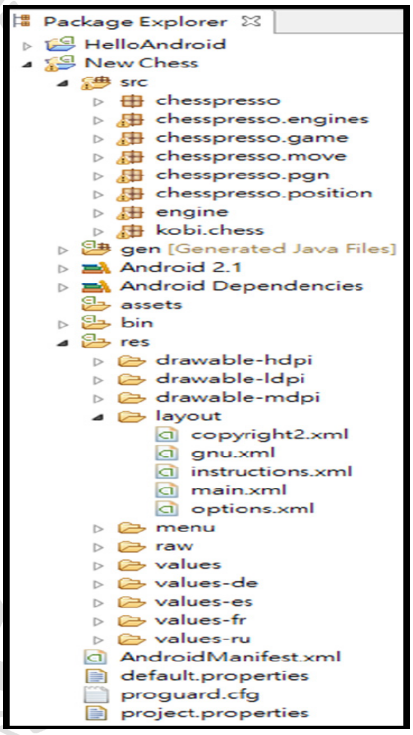

Figure 2-16

12. Then we can follow what was introduced in exp. 2 to execute Chess APP experiment through AVD or COS-100, as shown in Figure 2-17 and 2-18.

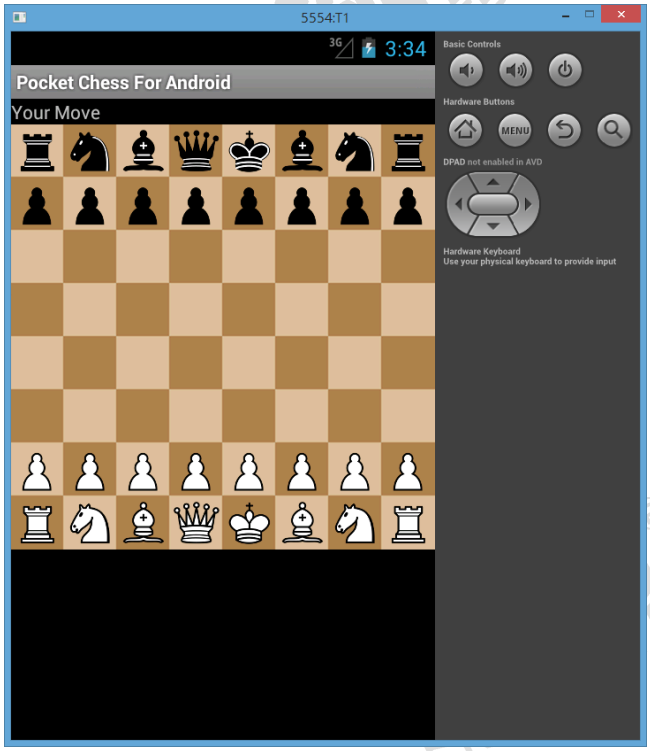

Figure 2-17

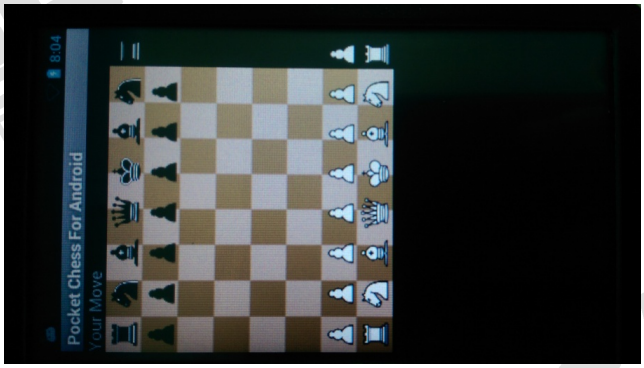

Figure 2-18

13. When executing, we will find that the direction of chessboard in COS-100 is wrong. To revise the program settings, open "AndroidManifest.xml" file, click "Application", and then revise each "Screen orientation" setting of Application Nodes, as shown in Figure 2-19. Sereen orientation select items:
    (1) Unspecified: (default)
    (2) Landscape: horizontal screen

(3) Portrai: vertical screen

(4) User: Using the user current settings

(5) Behind: the orientation setting of next activity

(6) Sensor: decide by sensor

(7) Nonsensor: like unspecified

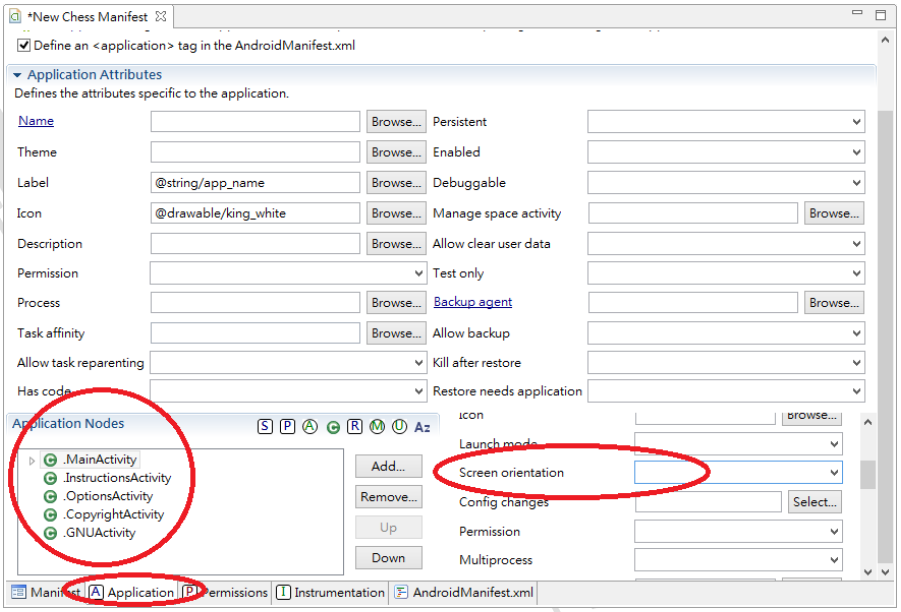

Figure 2-19

14. We can see that the background figure of chessboard is a little bit deviate, now we are going to adjust the size of it. The data stores in "New Chess" >> "res" >> "drawable-ldpi (/mdpi/ldpi…)", after adjusting the size, it can be shown normally, as shown in Figure 2-20.

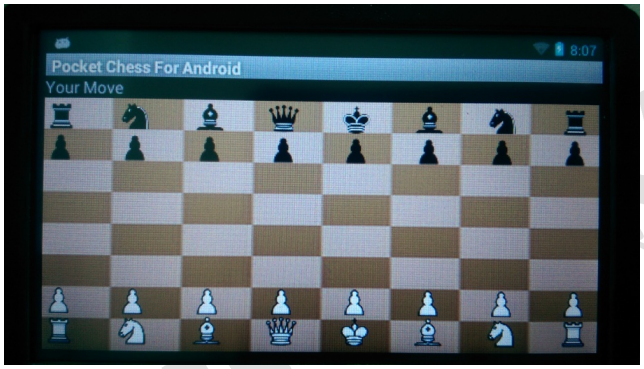

Figure 2-20

15. Then users can try to be familiar with the game and chess with computer, as shown in Figure 2-21.

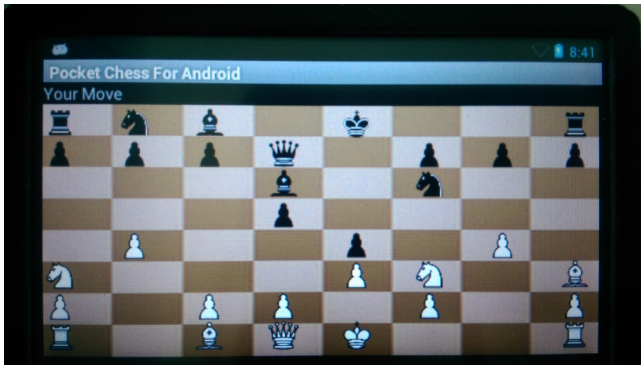

Figure 2-21

16. After understand the basic operation, users can also try to change the color of chessboard or the type of chess pieces, as shown in Figure 2-22.

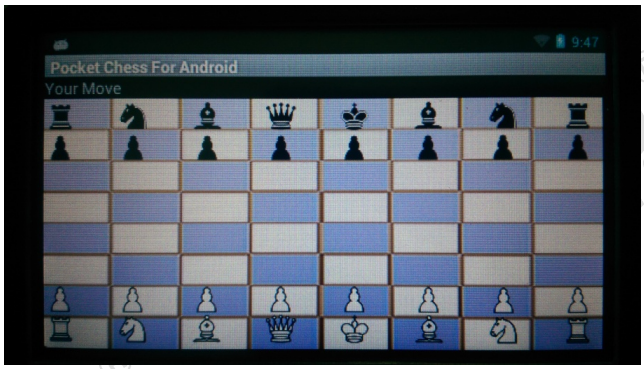

Figure 2-22

17. Now we are going to learn how to convert project to zip file. First click right on the project and then select "Export", as shown in Figure 2-23.

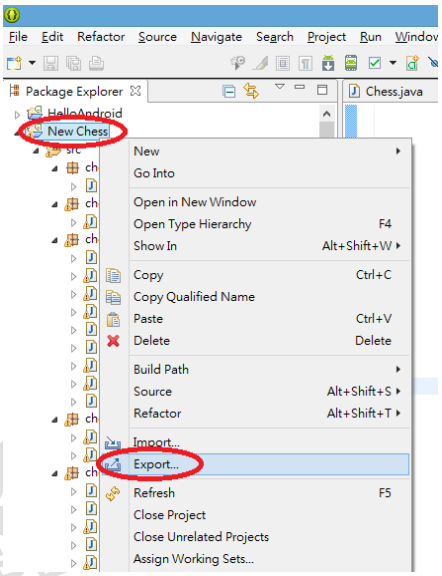

Figure 2-23

18. After Export window pop out, select "General >> Archive File", and then click "Next" to do next step, as shown in Figure 2-24.

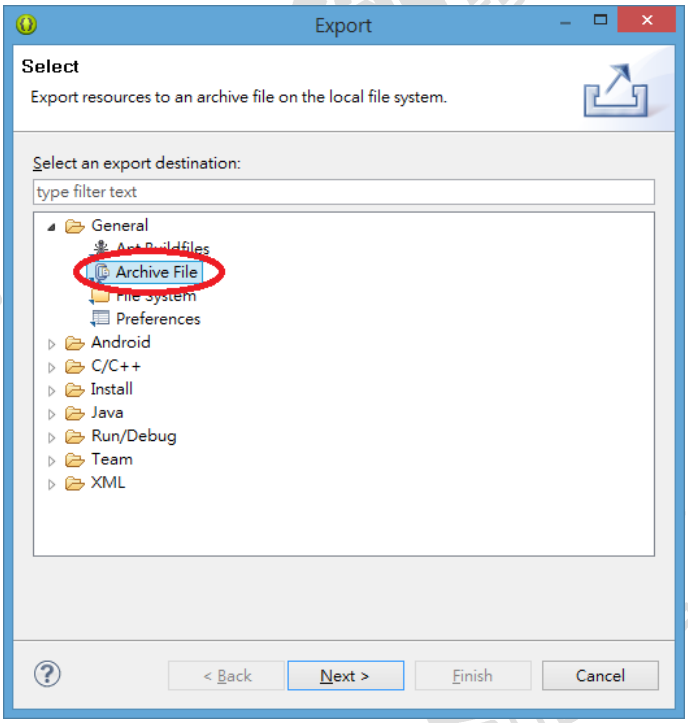

Figure 2-24

19. Select target project, click "Browse" and input the project zip file name and its storage path. After the setting is done, click "Finish", as shown in Figure 2-25.

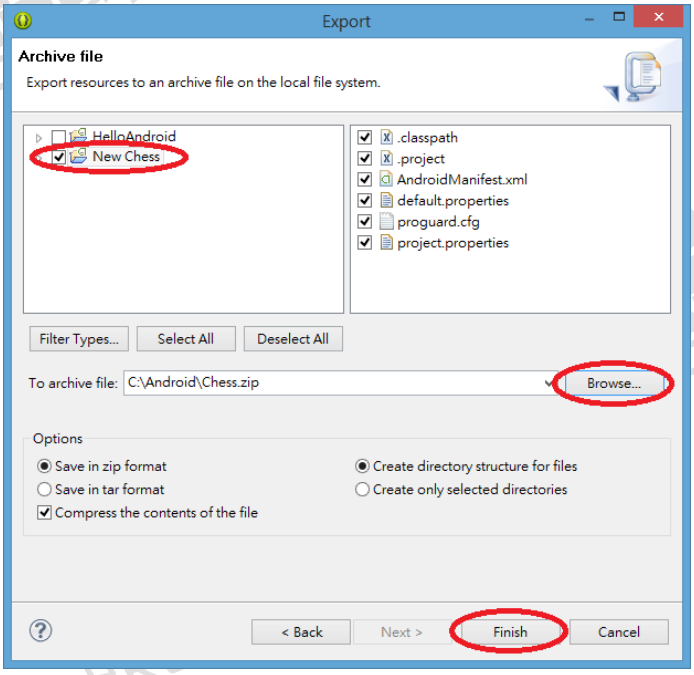

Figure 2-25

20. Then you can find the well-done project zip file at set path, as shown in Figure 2-26.

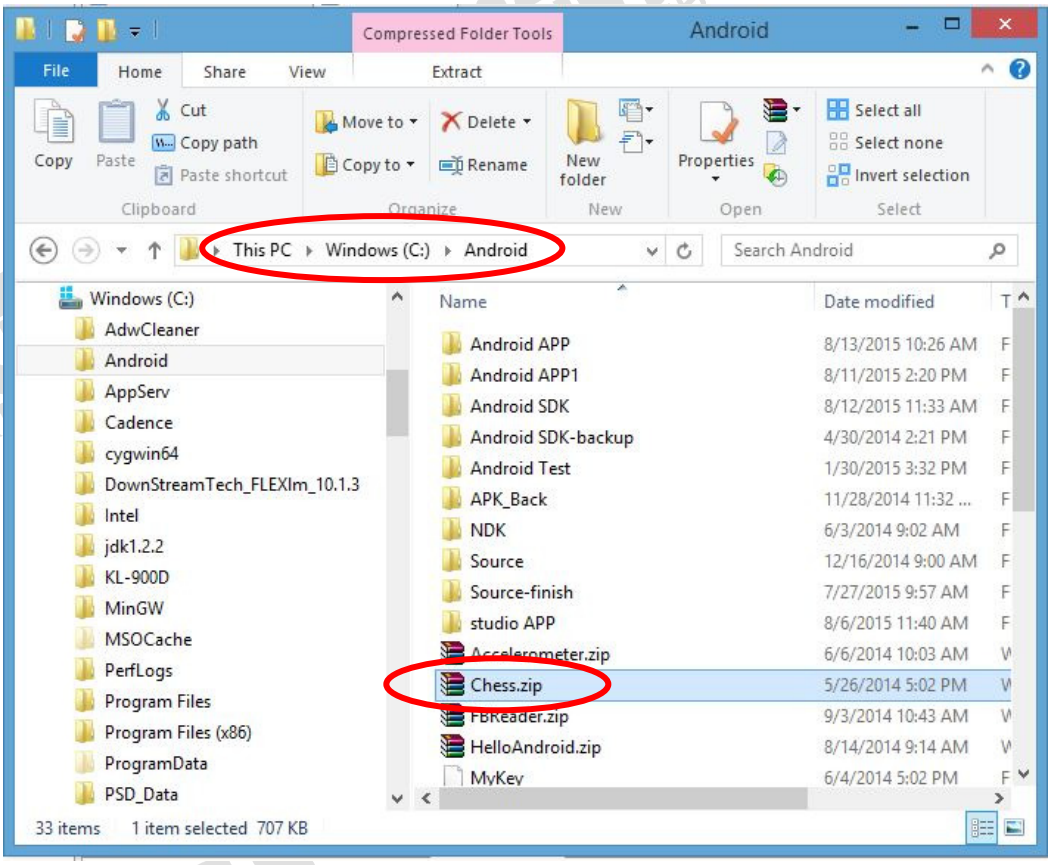

Figure 2-26

21. In later chapter we will teach users how to import project zip file to project.

2-4 APP Program Discussion

Program process overview: Main program is "src" >> "kobi.chess" >> "MainActivity.java", when program initial to setting "main.xml" for main form. And definition of "BoardView.java": (1) chessboard style, (2) the space between chess pieces, chess initial position, and draw the chessboard and chess etc.

"MainActivity.java" will be record the chess location coordinates, choose which one move, defintion Menu contents etc; the parts of the source code are as shown below.

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     // sets textView
     txtStatus = (TextView)this.findViewById(R.id.txtStatus);
     // sets view
     final BoardView boardView = (BoardView)findViewById(R.id.BoardView);
     ...
}

private synchronized void computerMove(){
    //Synchronized
    String temp = engine.go();
    this.boardView.displayPieces(engine.getBoard(ply));
    txtStatus.setText(R.string.activity_yourMove);
    if (engine.isMate()) // check computer move
    showToastNotification(this.getString(R.string.activity_mate));
    else if (engine.isCheck())
        showToastNotification(this.getString(R.string.activity_check));
    else if (engine.isDraw())
    showToastNotification(this.getString(R.string.activity_draw));
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
    case R.id.startGame:
        break;
    case R.id.Copyright:
        Intent myIntent3 = new Intent(this.getBaseContext(),
        CopyrightActivity.class);
        startActivityForResult(myIntent3, ACITIVITY_HELP);
        break;
    case R.id.help:
        Log.e("pointerSquare", "MENU_HELP");
        Intent myIntent2 = new Intent(this.getBaseContext(),
        InstructionsActivity.class);
        startActivityForResult(myIntent2, ACITIVITY_HELP);
        break;
    ...
    return super.onOptionsItemSelected(item);
}
```

1. **Chessboard style**

   As we discussed before, we can revise and adjust the size of background figure, the type of chessboard and also the type of chess pieces, but the question is: How to do that in program?

   In "res" >> "drawable-hdpi", there are all kinds of figures that show in APP, users are able to change the types of figure. Here we correspond through parts of the source code of "BoardView.java" to the name of each figure to display different results, as shown in Figure 2-27.

```
// Nwe Chess>>src>>kobi.chess>>BoardView.java
// set chessboard map
m_Board = BitmapFactory.decodeResource(getResources(),
R.drawable.chessboard);
// set white pieces & black pieces
```

```
m_ChessPieces = new Bitmap[12];
m_ChessPieces[pieces.get("K")] =
BitmapFactory.decodeResource(getResources(), R.drawable.king_white);
...
m_ChessPieces[pieces.get("k")] =
BitmapFactory.decodeResource(getResources(), R.drawable.king_black);
...
```

Figure 2-27

2. **The space between chess pieces**

To adjust the space between chess pieces, users can take part of the source code to do the revise through "BoardView.java", as shown in Figure 2-28.

```
// Nwe Chess>>src>>kobi.chess>>BoardView.java
@Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawBitmap(m_Board, 0, 0, null);

        // set pieces space size
        int pixelScaleWidth = ((this.getWidth()) / 6);
        int pixelScaleHeight = this.getHeight() / 8;
        for (int y=0; y<8; y++){
```

```
for (int x=0; x<8; x++){
    if (isMoveNow){
        if (!(startPoint.x == x && startPoint.y == 7 - y))
            actualDraw(canvas, pixelScaleWidth,
            pixelScaleHeight, y, x, 0, 0);
        else{
            int offsetX = (int) (movePoint.fx - startPoint.fx);
            int offsetY = (int) (movePoint.fy - startPoint.fy);
            actualDraw(canvas, pixelScaleWidth,
            pixelScaleHeight, y, x, offsetX, offsetY);
        }
    }
    else
    actualDraw(canvas, pixelScaleWidth, pixelScaleHeight, y, x,
    0, 0);
}}}
```

Figure 2-28

After understand the basic operation, students can think about how to enrich the functions. For example, students can make the available movement show around the chess piece when it is about to move or to revise the rule of movement, making your own Chess.