To get your feet off the ground with the project as quickly as possible, here is list of tasks you need to perform to configure Netbeans properly to launch a remote Linux Window Screen (called an "X11 window") onto your computer from my server. If you have a spotty internet connection, your X11 window may be a bit choppy.

INITIAL CONFIGURATION – THESE STEPS NEED TO BE PERFORMED ONLY ONE TIME

1.) Install **Xming** (Windows Users) or **Quartz** (Apple Users)

WINDOWS USERS: Install **Xming** your Windows laptop/machine. This is required to launch X11 windows (Linux/Unix based windows) on Windows machines. Go to:

http://sourceforge.net/projects/xming/

Click on the green download button and install it using the default options. When it launches the first time, allow it access. You may also want to pin it to your Taskbar so it easy to launch in the future.

APPLE USERS: Under `Finder > Applications > Utilities` there's a program for X11. Clicking it doesn't open up any programs, but instead takes you to an Apple Support page with the following:

"X11 is no longer included with OS X,
but X11 server and client libraries for OS X are available from the XQuartz project:

http://xquartz.macosforge.org/

You should use the latest available version of XQuartz"

Go to the website and install **XQuartz**. Use it to open a Terminal Window.

2.) Setup the Netbeans Environment.

ALL USERS: Create your project. Right click on the project, select "**Properties**". Click on "**C++ Compiler**". Under "**Include Directories**" add:

```
/usr/local/include
```

Under "**Additional Options**", add:

```
-L/usr/local/lib -lXCurses -std=c++11
```

Since the PDCurses (XCurses) library is not part of ANSI C/C++, we need to tell the compiler where to find the library when compiling your code.

Under "**Properties**" click "**Run**". Under "**Run Command**" append "**`lines [X] –cols [Y]`**", where **`[X]`** and **`[Y]`** represent the number of lines and columns for your window. I typically use 64 rows and 96 columns for this project, so the command line options I pass to the program are "**`–lines 64 –cols 96`**". The total "**Run Command**" should be:

```
"${OUTPUT_PATH}" –lines 64 –cols 96
```

RUNTIME CONFIGURATION – THESE STEPS NEED TO BE PERFORMED EVERY TIME YOU WORK

WINDOWS USERS:

1.) Launch Xming if you are Windows user.

2.) Launch Putty if you are a Windows user. BUT, before you login perform the following steps:

Under "**Host Name"** type "**`ezekiel.engr.csufresno.edu`**", as always.

Next, on the left hand column, double click on "**SSH**" near the bottom, to expand the tree. Then click on "**X11**". On the right side under "**X11 forwarding**" click on "**Enable X11 Forwarding**". Then click "**Open**" and type in your username and password to login.

APPLE USERS:

1.) Launch a Terminal Window from XQuartz.

2.) Login to the server using ssh, with X11 Forwarding Enabled.

```
ssh –X –lece71k[XX] ezekiel.engr.csufresno.edu
```

Replace [XX] with your login number.

ALL USERS:

3.) Once logged into your account with Putty or SSH, type:

```
echo $DISPLAY
```

Remember what is echoed back to you. Currently, when I type the command, it states "**`localhost:10.0`**", but this can change each time you log in.

LEAVE YOUR PUTTY OR SSH CONNECTION OPEN.
DO NOT CLOSE IT WHILE YOU CODE, DEBUG, AND RUN YOUR PROJECT.

Otherwise, you will not be able to launch the X11 window to your screen.

4. Launch Netbeans. Right click on your "**Project**", and click on "**Properties**". Then click on "**Run**", Under "**Environment**", click "**Add**". Under the "**Name**" box, add "**`DISPLAY`**". Under the "**Value**" box, add what the **`$DISPLAY`** variable echoed to you under Putty/SSH. For example, in my case, I added "**`localhost:10.0`**". Click "**OK**" twice. In the future, change it to whatever Putty/SSH says the **`$DISPLAY`** variable is after you login.

5. Code.

6. Compile and debug.

7. Run your code. Your program will compile on the server. Then, based on the values we've set, the program will launch an X11 window from the server onto your computer and run the game locally... assuming your code works.

INITIALIZATION FUNCTIONS FOR XCURSES

To get your feet off the ground, you will need to use the following functions:

```
Xinitscr()
PDC_set_title()
cbreak()
curs_set()
keypad()
noecho()
start_color()
resize_term()
nodelay()
```

Read about these functions in the documentation links I provided you!

http://pdcurses.sourceforge.net/doc/PDCurses.txt
http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/

Type in some example code. Create a screen and try to use the "`printw()`", "`move()`", "`mvprintw()`", and "`refresh()`" functions to print information to the standard screen. Whenever any of the XCurses functions are to be used in a source file, be sure to include the appropriate pre-processor directive:

```
#include <xcurses.h>
```

To close your window before exiting your program, use "`endwin();`". This will terminate your window properly. Call it before "`exit(EXIT_SUCCESS);`".

The aforementioned initialization functions should be placed into an `initXCurses()` function, and dropped into an `init.cpp` file. **THE FIRST LINE OF THE `init.cpp` FILE MUST BE:**

```
#define XCURSES
```

The following function definition for `initXCurses()` may be used:

```
//-----------------------------------------------------------//
void initXCurses(int argc, char** argv)
{
    Xinitscr(argc, argv);        // Initialize X Display Screen

    PDC_set_title("ECE 71 PONG BREAKOUT PROJECT");
                                 // Relabel Window Title
    cbreak();                    // Retain ^C Behavior

    curs_set(FALSE);             // Make Cursor Invisible
    keypad(stdscr, TRUE);        // Turn on Keypad
    noecho();                    // Don't echo input

    start_color();               // Start NCURSES Color

    nodelay(stdscr, TRUE);       // Do not wait for input

    if (getch()==KEY_RESIZE)     // Match screen to current
        resize_term(0,0);        // terminal size

    return;
}
//-----------------------------------------------------------//
```

To properly initialize the XCurses library to create an X11 window for your program, the main function should immediately call the **`initXCurses()`** function, and pass **`argc`** and **`argv`** into the function as arguments:

```
initXCurses(argc,argv);
```

In other words, place this statement into your main function as the first line of code.

INITIAL PROGRAMMING GOALS

Learning a new library from scratch can be daunting. Set the following goals:

1. Code/run a couple examples in the HOWTO links (use the `Xinitscr()` function instead of the `initscr()` function, however). Focus on the basic stuff – "`printw()`", "`move()`", and "`mvprintw()`". Print a few strings in different locations on your X11 screen. Call "`refresh()`" to update the screen so that information is moved from an internal buffer to the screen itself.

2. To keep the screen from immediately disappearing before terminating your program, at the end of your code change the behavior of the `nodelay()` function to wait for input via "`nodelay(stdscr, TRUE);`", call "`getch();`" to get a character, call "`endwin();`" to close the graphics window, and "`exit(EXIT_SUCCESS);`" end the program.

3. Use the PDCurses library documentation to figure out how to print a block to the screen using "`addch()`" with the `ACS_BLOCK` character.

4. Figure out how to center a line of text horizontally and/or vertically on the screen. The "`getmaxyx()`" function (which determines how many lines and columns your screen has), as well as "`mvprintw()`" and "`refresh()`" will be useful.

5. Print the paddle at the bottom of the screen.

6. Use a loop to get characters continuously, and check to see if KEY_LEFT, KEY_RIGHT, or KEY_DOWN was pressed for the left, right , and down arrow keys. If so, move the paddle appropriately.

7. Print the ball to the screen. Get it to move and reflect off the walls.

8. Figure out the deflection physics for the paddle. Get the ball to reflect off the paddle.

9. Print bricks to the screen. Allow the ball to interact with the bricks via deflection and destroying the bricks when the ball hits a brick.

10. Etc.