

# Torque Game Builder – Checkers Tutorial - Part 6

## 6. Moving Checkers

### 6.1 Moving a piece

Time to create our move code. First we need to add an attempt-to-move function in our “clientCheckers.cs” script file.

```
function attemptMovePiece(%x, %y)
{
    // check if the move is legal
    %legal = ClientCheckerBoard.isLegalMove($clientSelectedX, $clientSelectedY, %x, %y);

    // here we check if we are moving to the same position, if so we want to reset our piece
    if(%legal $= "same")
    {
        // we grab the tile's position
        %pos = ClientCheckerBoard.getTilePosition($clientSelectedX, $clientSelectedY);

        // we set the layer of the checker and its eyes
        $clientSelectedPiece.setLayer($layers::checkerLayer);
        // dismount the checker from the mouse object
        $clientSelectedPiece.dismount();
        // reset the checker's position to the original position
        $clientSelectedPiece.setPosition(%pos);

        // set selected to false
        $clientSelected = false;
    } else if(%legal)
    {
        // request a move from the server
        commandToServer('attemptMovePiece', %x, %y);
    }
}
```

**Code Sample 6.1.1**

Because we already have our isLegalMove() function created, this function is a lot smaller and tighter than it could potentially be. First we check if this move is legal by passing the selected positions and the attempted move to positions. The result is stored in %legal. First we check if the word “same” is passed back. If so then we just want to set the checkers position back to the same location without triggering a change in turns. Then we check to see if legal is true. If so we tell the server we're attempting to move the piece. Now we can add this server command to the “serverCommands.cs” file in the “server” folder.

```
function serverCmdAttemptMovePiece(%client, %x, %y)
{
    // pass the attempted move values on to the server function
    serverAttemptMovePiece(%client, %x, %y);
}
```

**Code Sample 6.1.2**

## Torque Game Builder – Checkers Tutorial - Part 6

This simply calls a normal function and passes the values. Add this new function to the “serverCheckers.cs” file. This function is rather large and is split up into two parts, here is the first part.

```
function serverAttemptMovePiece(%client, %x, %y)
{
    // grab whether the move is legal
    %legal = ServerCheckerBoard.isLegalMove($playersTurn.selectedX, $playersTurn.selectedY,
    %x, %y);
    // init jumped to false
    %jumped = false;

    // if we receive more than just a 0 or 1 for true or false then that's the position of a jumped piece
    if(getWord(%legal, 1) != "")
    {
        // divided up the legal response since this is now a piece location that was jumped
        %jumpedX = getWord(%legal, 0);
        %jumpedY = getWord(%legal, 1);

        // grab the jumped piece
        %jumpedPiece = ServerCheckerBoard.getPiece(%jumpedX, %jumpedY);

        // grab the other player
        %otherPlayer = getOtherPlayer(%client);

        // remove the piece
        ServerCheckerBoard.removePiece(%jumpedX, %jumpedY);

        // tell the client's to remove the piece
        commandToClient(%client, 'destroyPiece', %jumpedX, %jumpedY);
        commandToClient(%otherPlayer, 'destroyPiece', %jumpedX, %jumpedY);

        // decrement the proper count
        if(%jumpedPiece == $red || %jumpedPiece == $redKing)
        {
            $redCount--;
        } else
        {
            $blueCount--;
        }
    }

    // check if this means the game is over
    serverEndGameCheck();

    // toggle jumped to true
    %jumped = true;
}
```

First we check if the move is legal. We do the same thing we did on the ClientCheckerBoard with this ServerCheckerBoard. We also toggle a %jumped local variable to false. What we do at first is a little weird. What we are doing is grabbing the second space separated character. If this was a normal move then a true or false (a 1 or 0) is passed, so there wouldn't be a second

## Torque Game Builder – Checkers Tutorial - Part 6

character; however, if you remember in our `isLegalMove()` function if we jump a piece we pass back the position of the piece we jumped, so basically this is a check if a piece was jumped. If so we divide out the jumpedX and jumpedY positions, then grab the piece at that location with `getPiece()`. We get the otherPlayer and then remove the piece on the server. We then send a command to both clients to destroy the piece because it has been jumped. These are functions we need to create.

We then compare the jumped piece to figure out what team it is, we decrement the proper team's piece count, and then do a `serverEndGameCheck()`. This function will basically see if a team has lost. We end by toggling `%jumped` to true.

Here we continue with the second part of this function.

```
// check if the move is legal
if(%legal)
{
    // grab the type of piece that is selected
    %type = ServerCheckerBoard.getPiece($playersTurn.selectedX, $playersTurn.selectedY);

    // set the proper piece settings
    ServerCheckerBoard.setPiece($playersTurn.selectedX, $playersTurn.selectedY, $none);
    ServerCheckerBoard.setPiece(%x, %y, %type);

    // grab the other player
    %otherPlayer = getOtherPlayer(%client);

    // tell the client's to move the piece since this was a legal move
    commandToClient(%otherPlayer, 'moveCheckerPiece', %type, $playersTurn.selectedX,
$playersTurn.selectedY, %x, %y);
    commandToClient(%client, 'movePieceResponse', %x, %y, $yes);

    // reset the player's selection
    $playersTurn.selectedX = "";
    $playersTurn.selectedY = "";
    $playersTurn.hasSelected = false;

    // if this wasn't a double jump then we just swap turns
    swapTurns();
}
}
```

**Code Sample 6.1.3**

First we check if the move is legal. If so then we get the type of the selected checker piece, then set the piece of the selected spot to none and the piece of the new location to the piece selected. This will be triggered either when a player is simply moving one spot, or when a piece is jumped. The if will trigger as true as long as `%legal` isn't equal to 0. That means if it's a simple move that didn't trigger the previous if then it will still do this and it also means if `%legal` is a location (like "5 5") then it will trigger as well. This is exactly how we want it to work. After we set the piece locations we get the otherPlayer and send a command to both clients. To the otherPlayer (whose turn it isn't) we send a command to `moveCheckerPiece` and to the client (whose turn it is) we send a command to `movePieceResponse` passing the x, y, and a yes. We then reset the selection values and trigger the `swapTurns()` function.

## Torque Game Builder – Checkers Tutorial - Part 6

There are a few function that we called in this last function that we have yet to create. We will start with the client commands. Add these to your “clientCommands.cs” file.

```
function clientCmdMovePieceResponse(%x, %y, %answer)
{
    // here is the server's response to use trying to move a piece,
    // if its a no then we tell the client, if yes then we move it
    if(!%answer)
    {
        MessageBoxOK("You cannot move...", "You cannot move to the selected spot!", "");
    } else
    {
        clientMoveSelectedPiece(%x, %y);
    }
}
```

**Code Sample 6.1.4**

First we have the move piece response. If the answer is a no then we pop an OK box. If it's a yes then we call a function clientMoveSelectedPiece()... We will add this function to our “clientCheckers.cs” file.

```
function clientMoveSelectedPiece(%x, %y)
{
    // grab the tile's position from logical to world
    %pos = ClientCheckerBoard.getTilePosition(%x, %y);

    // dismount the selection from the mouse and set its position to
    // the new spot
    $clientSelectedPiece.dismount();
    $clientSelectedPiece.setPosition(%pos);

    // we set the layer of the checker and its eye's
    $clientSelectedPiece.setLayer($layers::checkerLayer);

    // grab the image and reset it, also set the new image
    %piece = ClientCheckerBoard.images[$clientSelectedX,$clientSelectedY];
    ClientCheckerBoard.images[$clientSelectedX,$clientSelectedY] = "";
    ClientCheckerBoard.images[%x,%y] = %piece;

    // grab the object's type
    %type = ClientCheckerBoard.getPiece($clientSelectedX, $clientSelectedY);

    // set the pieces position
    ClientCheckerBoard.setPiece($clientSelectedX, $clientSelectedY, $none);
    ClientCheckerBoard.setPiece(%x, %y, %type);

    // reset the client's selection
    $clientSelectedX = "";
    $clientSelectedY = "";
    $clientSelected = false;
}
```

**Code Sample 6.1.5**

## Torque Game Builder – Checkers Tutorial - Part 6

First we get the tile position of the logical position to move to. Then we dismount the selected piece and set its position to the position we just got. We set its layer to the checkers layer again. We grab the image of the selected location and store it, then we set the selected position's image to empty and the new position to the piece. We get the type of the selected piece (this is for the logical board and not the image board), then set the logical board at the selected piece's position to none and set the new position to the piece we're moving. Basically we just set our old positions to empty and our new positions to our piece. We then reset the selected values on the clients.

Now on to our next couple client commands. Add these to your "clientCommands.cs" file.

```
function clientCmdMoveCheckerPiece(%type, %fromX, %fromY, %x, %y)
{
    // this moves the piece directly to a new location
    clientMovePiece(%type, %fromX, %fromY, %x, %y);
}

function clientCmdDestroyPiece(%x, %y)
{
    // this destroy's a checker piece
    clientDestroyPiece(%x, %y);
}
```

**Code Sample 6.1.6**

The first function is called when another client has moved a piece and this client just needs to move the piece. As you can see we simply pass off the function to a normal one. The second function does the same thing but for the destroy piece function. Create these new functions in the "clientChecker.cs" file.

```
function clientMovePiece(%type, %fromX, %fromY, %x, %y)
{
    // grab the tile's position from logical to world
    %pos = ClientCheckerBoard.getTilePosition(%x, %y);

    // get the piece's image
    %piece = ClientCheckerBoard.images[%fromX, %fromY];

    // set the position of our piece
    %piece.setPosition(%pos);

    // reset the image in the previous slot and set the new one
    ClientCheckerBoard.images[$clientSelectedX,$clientSelectedY] = "";
    ClientCheckerBoard.images[%x,%y] = %piece;

    // reset the piece in the previous slot and set the new one
    ClientCheckerBoard.setPiece(%fromX, %fromY, $none);
    ClientCheckerBoard.setPiece(%x, %y, %type);
}
```

**Code Sample 6.1.7**

This function is just to move a piece from one location to the other, so we grab the actual position from the logical tile position, then we get the image of the piece to move, and we set its position

## Torque Game Builder – Checkers Tutorial - Part 6

to the new position. We then reset the image's array value of the old location to empty, then set the new location to the image of the moved piece. We then set the old logical location to empty and the new logical location to the type of the moved piece. Add our next function right after our previous function.

```
function clientDestroyPiece(%x, %y)
{
    // grab the team and the image data of the piece
    // to be destroyed
    %team = ClientCheckerBoard.getPiece(%x, %y);
    %image = ClientCheckerBoard.images[%x, %y];

    // check which team the piece is
    if(%team == $red)
    {
        // remove the piece from the team's container
        ClientCheckerBoard.redPieces.remove(%image);
    } else if(%team == $blue)
    {
        // remove the piece from the team's container
        ClientCheckerBoard.bluePieces.remove(%image);
    }

    // remove the piece image from the image data
    ClientCheckerBoard.images[%x, %y] = "";

    // remove the piece from the board data
    ClientCheckerBoard.removePiece(%x, %y);

    %image.safeDelete();
}
```

### **Code Sample 6.1.8**

This will handle when a checker piece needs destroyed. We grab the logical piece and the image piece, then check what team the piece is from and remove it properly from that team's data containers. We then set its image location to empty and call `removePiece()` on its logical location. We end by calling `safeDelete()` on the piece.

We have one final function to add before we can test this. We need to add the `swapTurns()` function to our "serverCheckers.cs" file in the "server" folder.

```
function swapTurns()
{
    // get the other player
    %player = getOtherPlayer($playersTurn);

    // set the turn to the other player
    setTurn(%player);
}
```

### **Code Sample 6.1.9**

This function is very simple. We get the `otherPlayer` and call `setTurn()` on that player...

# Torque Game Builder – Checkers Tutorial - Part 6

## 6.2 Test it!

Do the usual process of firing up the first instance of TGB and starting the server, then fire up the second instance and join the server. Now bounce back and forth and try and play the game. It should work: you should also be able to jump a piece! On top of that, both clients should be properly mirroring each other.



## Conclusion

I left a plug for an end game check function. You could create that function and check if any team is out of pieces and trigger a response... I also left a couple other things open, like creating kings and double jumping, but now you know how to set up the networking and game logic to set this up. You can also use the checkers demo scripts as a reference. You will notice a lot of similarities between this tutorial and the checkers demo.

This is the first TGB demo and tutorial with networking, I hope you learned a lot, had a lot of fun, and now understand the turn based TGB networking :)

Best wishes,

Matthew Langley, the TGB team, and the GarageGames team.