

# Torque Game Builder – Checkers Tutorial - Part 5

## 5. Checker Selection

### 5.1 Creating a selection system

We will start by creating our selection system. For this we need to add some mouse callbacks, which will go in our presently empty mouse.cs, so add this to your mouse.cs.

```
function SceneWindow2D::onMouseMove(%this, %modifier, %worldPos, %mouseClicks)
{
    // set the position of the object that is designated to follow the mouse
    $mouseObj.setPosition(%worldPos);
}

function SceneWindow2D::onMouseDragged(%this, %modifier, %worldPos, %mouseClicks)
{
    // set the position of the object that is designated to follow the mouse
    $mouseObj.setPosition(%worldPos);
}
```

**Code Sample 5.1.1**

This is the first of two mouse callbacks. It is called when the mouse moves and when it is dragged. Here we simply set the position of our mouse object to the position our mouse moves. Now add the next one.

```
function SceneWindow2D::onMouseUp(%this, %modifier, %worldPos, %mouseClicks)
{
    // if the clientcheckerboard is created then pick the tile
    if(isObject(CheckerTileLayer))
        %tile = CheckerTileLayer.pickTile(%worldPos);

    // if tile has a value then attempt to select it
    if(%tile != "")
        attemptSelectChecker(getWord(%tile, 0), getWord(%tile, 1));
}
```

**Code Sample 5.1.2**

This happens when the mouse is up, so when the user clicks and lets go, first we perform a pickTile on the location clicked, if the boardLayer exists. Doing the check for a boardLayer prevents us getting console spam before our boardLayer is created. Next, we make sure that an object is returned. If so we attempt to select that position that we clicked. Now lets create the attempt-to-select-function. Add this to your “clientCheckers.cs” in your “client” folder:

```
function attemptSelectChecker(%x, %y)
{
    // first we check if it is fact our turn
    if(!$myTurn)
```

## Torque Game Builder – Checkers Tutorial - Part 5

```
{
} else if($clientSelected)
{
// this means we already have a checker selected, so lets call our
// attempt to move function
attemptMovePiece(%x, %y);
} else
{
// grab the selected piece
%selection = ClientCheckerBoard.getPiece(%x, %y);

// get the possible king piece for that team
%king = getKing($playerTeam);

// if it's of the right team then let's attempt to check it
if(%selection == $playerTeam || %selection == %king)
{
// ask the server if we can select it
commandToServer('attemptSelectChecker', %x, %y);
} else
{
// let the console know this is an invalid selection
echo("invalid selection");
}
}
}
```

**Code Sample 5.1.3**

First we check to make sure it is our turn. If it isn't, we do nothing. Then we check if \$clientSelected is already triggered to true. If so that means this is an attempted move. We then pass off to our attempt move function. Finally we end with our else which means it is our turn and we haven't already selected something, so this means we must be attempting to select something, so we grab the selection with getPiece() and then the king. We then compare our selection to \$playerTeam (which represents a normal piece of that team) and the %king. If our selection is valid as either a normal piece or a king piece we then send an attempt to request to the server. If the piece you are trying to select isn't either a normal piece or a king of your team then you echo out an invalid selection.

Now lets add the serverCmd function to the "serverCommands.cs" in the "server" directory.

```
function serverCmdAttemptSelectChecker(%client, %x, %y)
{
// pass the attempted selection values on to the server function
serverAttemptSelectChecker(%client, %x, %y);
}
```

**Code Sample 5.1.4**

We simply pass off the passed values to a normal function. Add the normal function to your serverCheckers.cs.

## Torque Game Builder – Checkers Tutorial - Part 5

```
function serverAttemptSelectChecker(%client, %x, %y)
{
    // check if it's this client's turn
    if($playersTurn != %client)
    {
        // it is not this client's turn, clever client, the server is all knowing
        commandToClient(%client, 'notYourTurn');
    } else if($playersTurn == %client)
    {
        // grab the selection
        %selection = ServerCheckerBoard.getPiece(%x, %y);

        // grab the possible king
        %king = getKing(%client.team);

        // check if this is the client's piece, we also check if
        // it's on of their kings
        if(%selection == %client.team || %selection == %king)
        {
            // proper move, process sit
            serverSelectChecker(%client, %x, %y);
        } else
        {
            // the client cannot move another team's piece, bad client
            commandToClient(%client, 'selectCheckerResponse', %x, %y, $no);
        }
    }
}
```

**Code Sample 5.1.5**

Here we first check to ensure that the current player's turn is actually this client requesting a turn. If not we sent a command back for “not your turn”. We will need to add this command to the client commands. If it is that client's turn then we grab the selection the client is trying to select, then we grab the king of that team. We then do the same comparison we did on the client and check to make sure its of the same team or the king of the same team. If so we call `serverSelectChecker`; if not we send a response back to the client with a `commandToClient` passing the x, y, and a no.

Now let's add the `serverSelectChecker()` function to the same file after our previous function.

```
function serverSelectChecker(%client, %x, %y)
{
    // set the selected piece
    $playersTurn.selectedX = %x;
    $playersTurn.selectedY = %y;
    $playersTurn.hasSelected = true;

    // respond to the client
    commandToClient(%client, 'selectCheckerResponse', %x, %y, $yes);
}
```

**Code Sample 5.1.6**

## Torque Game Builder – Checkers Tutorial - Part 5

This function is called when we are sure the client can select the checker piece. We set a few values on the client, the selectedX and selectedY and then a bool for hasSelected. We end with a call to the client passing the x, y, and a yes this time. Let's add our client commands to the "clientCommands.cs" file in the "client" folder.

```
function clientCmdNotYourTurn()
{
    // tell the client that you -cannot- play when its NOT your turn ;)
    MessageBoxOK("Not your turn...", "It is currently not your turn", "");
}

function clientCmdSelectCheckerResponse(%x, %y, %answer)
{
    // here is our select response, if its a no we tell the client that you cannot
    // select the piece, if its a yes then we select the piece
    if(!%answer)
    {
        MessageBoxOK("Invalid Selection...", "You cannot select this checker piece!", "");
    } else
    {
        clientSelectChecker(%x, %y);
    }
}
```

### Code Sample 5.1.7

In our first function we simply pop a message box letting the player know it's not currently their turn. The second function is the response to the client attempting to select a piece. If the answer is a no then we pop up an OK box telling them so. If it is valid then we call a clientSelectChecker() function... let's add that function to our clientCheckers.cs.

```
function clientSelectChecker(%x, %y)
{
    // set the selection with what's passed in
    $clientSelectedX = %x;
    $clientSelectedY = %y;
    $clientSelected = true;

    // call the mount function to mount the checker to the mouse
    ClientCheckerBoard.mountCheckerToMouse(%x, %y);
}
```

### Code Sample 5.1.8

We pass in the x and y of the piece to be selected and we set three global variables: one to store the x, one to store the y, and the last is a bool to store that it has selected a checker piece. We finish off by calling a ClientCheckerBoard.mountCheckerToMouse() function passing the x and y. Add this function after our previous function.

```
function ClientCheckerBoard::mountCheckerToMouse(%this, %x, %y)
{
    // grab the checker
```

# Torque Game Builder – Checkers Tutorial - Part 5

```
%checker = %this.images[%x, %y];  
  
// store the selected checker  
$clientSelectedPiece = %checker;  
  
// change the layers of the checker and its eyes so we can see  
// it over other pieces, while mounted  
%checker.setLayer(0);  
  
// mount the piece to the mouse  
%checker.mount($mouseObj);  
}
```

*Code Sample 5.1.9*

First we grab the image of the checker selected, then we store that as the `$clientSelectedPiece`. We then set that checker's layer to 0, that way it will be completely visible, and finally we mount that checker piece to the object following the mouse's position.

## 5.2 Test it!

Now we can test this. Fire up your first instance and start your server as usual, then fire up the second instance and join the server as usual. Now with your checkerboards up go to the first client and try and select one of the moon pieces that are *not* of your team. It shouldn't do anything. Now try and select one of the sun pieces. It should properly select it and mount it to the mouse. You can move it around!



*Figure 5.2.1*