```
    1: // Copyright 2023 Thomas O'Connor
    2: #include "Checkers.hpp"
    3:
    4: // Interactors
    5: void Checkers::selectPiece(sf::Vector2i mouseLocation) {
    6:     // calculate the coordiate pair in relation to the 2D char array
    7:     int R = 0, B = 0;
    8:     for (int k = 0; k < 2; k++) {
    9:         for (int i = 0; i < BOARD_DIMENSIONS; i++) {
   10:             for (int j = 0; j < BOARD_DIMENSIONS; j++) {
   11:                 if (k == 0) {
   12:                     if (currentGameState[i][j] == 'r') R++;
   13:                     if (currentGameState[i][j] == 'b') B++;
   14:                 } else {
   15:                     if (currentGameState[i][j] == 'w' && R == 11) {
   16:                         currentGameState[i][j] = 'r';
   17:                         break;
   18:                     } else if (currentGameState[i][j] == 'w' && B == 11)
{
   19:                         currentGameState[i][j] = 'b';
   20:                         break;
   21:                     }
   22:                 }
   23:             }
   24:         }
   25:     }
   26:     sf::Vector2i arenaLocation((mouseLocation.x - 32) / 64, (mouseLocatio
n.y - 32) / 64);
   27:     if (playerTurn) {
   28:         // red turn
   29:         if (currentGameState[arenaLocation.y][arenaLocation.x] == 'r') {
   30:             currentGameState[arenaLocation.y][arenaLocation.x] = 'w';
   31:             return;
   32:         }
   33:     } else {
   34:         // black turn
   35:         if (currentGameState[arenaLocation.y][arenaLocation.x] == 'b') {
   36:             currentGameState[arenaLocation.y][arenaLocation.x] = 'w';
   37:             return;
   38:         }
   39:     }
   40: }
   41:
   42: // Draw game in SFML
   43: void Checkers::draw(sf::RenderTarget& target, sf::RenderStates states) co
nst {
   44:     // Load piece textures
   45:     sf::Texture blackPiece, redPiece, whitePiece, woodBacking;
   46:     if (!blackPiece.loadFromFile("checkers/blackpawn.png")) exit(1);
   47:     if (!redPiece.loadFromFile("checkers/redpawn.png")) exit(1);
   48:     if (!whitePiece.loadFromFile("checkers/whitepawn.png")) exit(1);
   49:     if (!woodBacking.loadFromFile("checkers/wood_texture.png")) exit(1);
   50:     sf::Sprite WB;
   51:     WB.setTexture(woodBacking);
   52:     WB.setScale(1, 1.2);
   53:     target.draw(WB);
   54:     for (int i = 0; i < BOARD_DIMENSIONS; i++) {
   55:         for (int j = 0; j < BOARD_DIMENSIONS; j++) {
   56:             // Draw states stored in currentGameState
   57:             switch (currentGameState[i][j]) {
   58:                 case '.':
   59:                 {
   60:                     sf::RectangleShape R;
   61:                     R.setSize(Vector2f(TILE_SIZE, TILE_SIZE));
   62:                     R.setFillColor(sf::Color::Red);
```

```
   63:                     R.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_S
IZE + TILE_SIZE/2);
   64:                     target.draw(R);
   65:                     break;
   66:                 }
   67:                 case 'p':
   68:                 {
   69:                     sf::RectangleShape B;
   70:                     B.setSize(Vector2f(TILE_SIZE, TILE_SIZE));
   71:                     B.setFillColor(sf::Color::Black);
   72:                     B.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_S
IZE + TILE_SIZE/2);
   73:                     target.draw(B);
   74:                     break;
   75:                 }
   76:                 case 'r':
   77:                 {
   78:                     sf::RectangleShape B;
   79:                     B.setSize(Vector2f(TILE_SIZE, TILE_SIZE));
   80:                     B.setFillColor(sf::Color::Black);
   81:                     B.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_S
IZE + TILE_SIZE/2);
   82:                     target.draw(B);
   83:                     sf::Sprite RP;
   84:                     RP.setTexture(redPiece);
   85:                     RP.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_
SIZE + TILE_SIZE/2);
   86:                     target.draw(RP);
   87:                     break;
   88:                 }
   89:                 case 'b':
   90:                 {
   91:                     sf::RectangleShape B;
   92:                     B.setSize(Vector2f(TILE_SIZE, TILE_SIZE));
   93:                     B.setFillColor(sf::Color::Black);
   94:                     B.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_S
IZE + TILE_SIZE/2);
   95:                     target.draw(B);
   96:                     sf::Sprite BP;
   97:                     BP.setTexture(blackPiece);
   98:                     BP.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_
SIZE + TILE_SIZE/2);
   99:                     target.draw(BP);
  100:                     break;
  101:                 }
  102:                 case 'w':
  103:                 {
  104:                     sf::RectangleShape B;
  105:                     B.setSize(Vector2f(TILE_SIZE, TILE_SIZE));
  106:                     B.setFillColor(sf::Color::Black);
  107:                     B.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_S
IZE + TILE_SIZE/2);
  108:                     target.draw(B);
  109:                     sf::Sprite WP;
  110:                     WP.setTexture(whitePiece);
  111:                     WP.setScale(0.6, 0.6);
  112:                     WP.setPosition(j * TILE_SIZE + TILE_SIZE/2, i * TILE_
SIZE + TILE_SIZE/2);
  113:                     target.draw(WP);
  114:                     break;
  115:                 }
  116:             }
  117:         }
  118:     }
  119:     sf::Font font;
```

```
  120:     font.loadFromFile("checkers/arial.ttf");
  121:     sf::Text turnText("", font, 40);
  122:     turnText.setFillColor(sf::Color::White);
  123:     turnText.setPosition(Vector2f(10, 0));
  124:     if (playerTurn) {
  125:         // Red turn
  126:         turnText.setString("R");
  127:     } else {
  128:         // Black turn
  129:         turnText.setString("B");
  130:     }
  131:     target.draw(turnText);
  132: }
  133:
  134: // Initialize game storage vectors
  135: void Checkers::initializeBase() {
  136:     currentGameState.resize(BOARD_DIMENSIONS);
  137:     for (int i = 0; i < BOARD_DIMENSIONS; i++) {
  138:         currentGameState[i].resize(BOARD_DIMENSIONS);
  139:         for (int j = 0; j < BOARD_DIMENSIONS; j++) {
  140:             // Draw red background tiles
  141:             if (((j + (i%2))%2)) {
  142:                 if (i <= 2) {
  143:                     // Black piece represented as 'b' (black)
  144:                     currentGameState[i][j] = 'b';
  145:                 } else if (i >= 5) {
  146:                     // Red piece represented as 'r' (red)
  147:                     currentGameState[i][j] = 'r';
  148:                 } else {
  149:                     // Black background tile represented 'p' (playable)
  150:                     currentGameState[i][j] = 'p';
  151:                 }
  152:             } else {
  153:                 // Red background tile represented as '.' (invalid)
  154:                 currentGameState[i][j] = '.';
  155:             }
  156:         }
  157:     }
  158: }
  159:
  160: bool mouseInGameBounds(sf::Vector2i mouseLocation) {
  161:     if ((mouseLocation.x >= 32 && mouseLocation.x <= TILE_SIZE * 8 + 32)
&&
  162:     (mouseLocation.y >= 32 && mouseLocation.y <= TILE_SIZE * 8 + 32)) ret
urn 1;
  163:     return 0;
  164: }
```