Light

//playerLight.setPosition(player->getPosition() + (player->getDimensions() / 2.0f) - (light.getDimensions() / 2.0f));

/\*glm::vec2 mouseCoords = camera.convertScreenToWorld(inputManager.getMouseCoords());

light.setPosition(glm::vec2(mouseCoords.x - 4 \* UNIT\_WIDTH, mouseCoords.y - 4 \* UNIT\_HEIGHT));

return;

Color color = light.getColor();

timer += frameTime;

// approximately every 5 seconds turn off / on

if (timer >= MILISECONDS \* 0.05f) {

if (flip) {

alpha = color.a - 2.25f;

if (alpha <= 0) {

light.setColor(color.r, color.g, color.b, 0);

flip = false;

}

else {

light.setColor(color.r, color.g, color.b, alpha);

}

}

else {

alpha = color.a + 2.25f;

if (alpha >= 255) {

light.setColor(color.r, color.g, color.b, 255);

flip = true;

}

else {

light.setColor(color.r, color.g, color.b, alpha);

}

}

timer = 0.0f;

} \*/

//// node at upper rigth corner

//if ((row - 1 >= 0) && (column + 1 < columnNumber) && !searchSpace[row - 1][column + 1].isBlock()) {

// Node temp = searchSpace[row - 1][column + 1];

// temp.setG(node.getG() + 14);

// temp.setH(manhattanHeuristics(temp, \*finalNode));

// neighbors.push\_back(temp);

//}

//// node at bottom right corner

//if ((row + 1 < rowNumber) && (column + 1 < columnNumber) && !searchSpace[row + 1][column + 1].isBlock()) {

// Node temp = searchSpace[row + 1][column + 1];

// temp.setG(node.getG() + 14);

// temp.setH(manhattanHeuristics(temp, \*finalNode));

// neighbors.push\_back(temp);

//}

//// node at upper left corner

//if ((row - 1 >= 0) && (column - 1 >= 0) && !searchSpace[row - 1][column - 1].isBlock()) {

// Node temp = searchSpace[row - 1][column - 1];

// temp.setG(node.getG() + 14);

// temp.setH(manhattanHeuristics(temp, \*finalNode));

// neighbors.push\_back(temp);

//}

//// node at bottom left corner

//if ((row + 1 < rowNumber) && (column - 1 >= 0) && !searchSpace[row + 1][column - 1].isBlock()) {

// Node temp = searchSpace[row + 1][column - 1];

// temp.setG(node.getG() + 14);

// temp.setH(manhattanHeuristics(temp, \*finalNode));

// neighbors.push\_back(temp);

//}

//std::vector<Node\*> AStarAlgorithm::getAllNeighbors(Node\* node) {

// std::vector<Node\*> neighbors;

//

// int row = node->getRowIndex();

// int column = node->getColumnIndex();

//

// // node above

// if ((row - 1 >= 0) && !searchSpace[row - 1][column].isBlock()) {

// Node\* temp = &searchSpace[row - 1][column];

//

// temp->setG(node->getG() + 10);

// temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// neighbors.push\_back(temp);

// }

//

// // node below

// if ((row + 1 < rowNumber) && !searchSpace[row + 1][column].isBlock()) {

// Node\* temp = &searchSpace[row + 1][column];

//

// temp->setG(node->getG() + 10);

// temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// neighbors.push\_back(temp);

// }

//

// // node to the left

// if ((column - 1 >= 0) && !searchSpace[row][column - 1].isBlock()) {

// Node\* temp = &searchSpace[row][column - 1];

//

// temp->setG(node->getG() + 10);

// temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// neighbors.push\_back(temp);

// }

//

// // node to the right

// if ((column + 1 < columnNumber) && !searchSpace[row][column + 1].isBlock()) {

// Node\* temp = &searchSpace[row][column + 1];

//

// temp->setG(node->getG() + 10);

// temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// neighbors.push\_back(temp);

// }

//

// //// node at upper right corner

// //if ((row - 1 >= 0) && (column + 1 < columnNumber) && !searchSpace[row - 1][column].isBlock() && !searchSpace[row][column + 1].isBlock() && !searchSpace[row - 1][column + 1].isBlock()) {

// // Node\* temp = &searchSpace[row - 1][column + 1];

//

// // temp->setG(node->getG() + 14);

// // temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// // neighbors.push\_back(temp);

// //}

//

// //// node at bottom right corner

// //if ((row + 1 < rowNumber) && (column + 1 < columnNumber) && !searchSpace[row + 1][column].isBlock() && !searchSpace[row][column + 1].isBlock() && !searchSpace[row + 1][column + 1].isBlock()) {

// // Node\* temp = &searchSpace[row + 1][column + 1];

//

// // temp->setG(node->getG() + 14);

// // temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// // neighbors.push\_back(temp);

// //}

//

// //// node at upper left corner

// //if ((row - 1 >= 0) && (column - 1 >= 0) && !searchSpace[row - 1][column].isBlock() && !searchSpace[row][column - 1].isBlock() && !searchSpace[row - 1][column - 1].isBlock()) {

// // Node\* temp = &searchSpace[row - 1][column - 1];

//

// // temp->setG(node->getG() + 14);

// // temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// // neighbors.push\_back(temp);

// //}

//

// //// node at bottom left corner

// //if ((row + 1 < rowNumber) && (column - 1 >= 0) && !searchSpace[row + 1][column].isBlock() && !searchSpace[row][column - 1].isBlock() && !searchSpace[row + 1][column - 1].isBlock()) {

// // Node\* temp = &searchSpace[row + 1][column - 1];

//

// // temp->setG(node->getG() + 14);

// // temp->setH(manhattanHeuristics(\*temp, \*finalNode));

//

// // neighbors.push\_back(temp);

// //}

//

// return neighbors;

//}

//int x = pow(node2.getRowIndex() - node1.getRowIndex(), 2);

//int y = pow(node2.getColumnIndex() - node1.getColumnIndex(), 2);

//return (abs(node2.getRowIndex() - node1.getRowIndex()) + abs(node2.getColumnIndex() - node1.getColumnIndex())) \* 10;

//return sqrt(x + y);

/\*GLint visionRadiusLocation = visionProgram.getUniformValueLocation("visionRadius");

GLint visionCenterLocation = visionProgram.getUniformValueLocation("visionCenter");

for (size\_t i = 0; i < lights.size(); i++) {

Light\* light = lights[i];

lightRadius = light->getWidth();

lightSource = light->getVisionCenter();

glUniform1f(visionRadiusLocation, lightRadius);

glUniform2f(visionCenterLocation, lightSource.x, lightSource.y);

std::vector<GLSL\_Object> visionVector = visibleObjects[light->getID()];

for (size\_t i = 0; i < visionVector.size(); i++) {

GLSL\_Object object = visionVector[i];

glDrawArrays(object.getMode(), object.getOffset(), object.getVertexNumber());

}

}

for (auto& x : visibleObjects) {

x.second.clear();

}\*/

/\*if (inputManager.isKeyPressed(SDLK\_a)) {

if (!checkCollision(player->getX() - PLAYER\_SPEED \* deltaTime, player->getY())) {

player->update(-PLAYER\_SPEED, 0.0f, deltaTime);

camera.setPosition(getCameraPosition(glm::vec2(player->getX(), player->getY())));

}

}

if (inputManager.isKeyPressed(SDLK\_d)) {

if (!checkCollision(player->getX() + PLAYER\_SPEED \* deltaTime, player->getY())) {

player->update(PLAYER\_SPEED, 0.0f, deltaTime);

camera.setPosition((getCameraPosition(glm::vec2(player->getX(), player->getY()))));

}

}

if (inputManager.isKeyPressed(SDLK\_w)) {

if (!checkCollision(player->getX(), player->getY() + PLAYER\_SPEED \* deltaTime)) {

player->update(0.0f, PLAYER\_SPEED, deltaTime);

camera.setPosition(getCameraPosition(glm::vec2(player->getX(), player->getY())));

}

}

if (inputManager.isKeyPressed(SDLK\_s)) {

if (!checkCollision(player->getX(), player->getY() - PLAYER\_SPEED \* deltaTime)) {

player->update(0.0f, -PLAYER\_SPEED, deltaTime);

camera.setPosition((getCameraPosition(glm::vec2(player->getX(), player->getY()))));

}

}\*/