Unit Testing Report

**MapTests/DefaultConstructor** is a test which checks that the nodes created when a map is constructed are not null.

**Input:**

Map m = Map();

**Expected Output:**

for (int x = 0; x < MAP\_COLUMNS; x++) {

for (int y = 0; y < MAP\_ROWS; y++) {

Assert::IsNotNull(m.nodes[x][y]);

}

}

**Output:**

None of the nodes are null.

**MapTests/AddCritter** is a test which checks if the map’s method for checking collisions and adding a critter to the appropriate node adds a critter to the appropriate node.

**Input:**

Map m = Map();

Critter c = Critter();

c.Init(Vector2{ 15, 15 }, Vector2{ 0, 0 });

m.Collisions(&c);

size\_t tx = (size\_t)(c.GetX() / (float)NODE\_WIDTH);

size\_t ty = (size\_t)(c.GetY() / (float)NODE\_HEIGHT);

Node\* n = m.nodes[tx][ty];

**Expected Output:**

Assert::AreEqual(n->critters.size(), (size\_t)1);

**Output:**

The amount of critters in node [tx][ty] is 1.

**NodeTests/DefaultConstructor** is a test which checks that when a node is created using the default constructor, its bounds are zero.

**Input:**

Node n = Node();

**Expected Output:**

Assert::AreEqual(n.bounds.min.x, 0.0f);

Assert::AreEqual(n.bounds.min.y, 0.0f);

Assert::AreEqual(n.bounds.max.x, 0.0f);

Assert::AreEqual(n.bounds.max.y, 0.0f);

**Output:**

n.bounds.min.x is 0.

n.bounds.min.y is 0.

n.bounds.max.x is 0.

n.bounds.max.y is 0.

**NodeTests/BoundsConstructor** is a test which checks that the bounds assigned to the node when it is constructed are correct.

**Input:**

Node n = Node(AABB{ Vector2{1, 2}, Vector2{3, 4} });

**Expected Output:**

Assert::AreEqual(n.bounds.min.x, 1.0f);

Assert::AreEqual(n.bounds.min.y, 2.0f);

Assert::AreEqual(n.bounds.max.x, 3.0f);

Assert::AreEqual(n.bounds.max.y, 4.0f);

**Output:**

n.bounds.min.x is 1.

n.bounds.min.y is 2.

n.bounds.max.x is 3.

n.bounds.max.y is 4.

Optimisation Diagnostics

A screen shot of a computer

Description automatically generatedI assessed performance mostly through Raylib’s DrawFPS() function, which was uncapped thanks to not setting a target FPS. I also looked at the diagnostics in visual studio, but they didn’t change much when I implemented my optimisations.

A screenshot of a computer

Description automatically generated

Above: VS Diagnostics of the original CCDS\_Optimise.

Below: Raylib FPS of the original CCDS\_Optimise.

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedAbove: VS Diagnostics after my optimisations.

Below: Raylib FPS after my optimisations.

A screenshot of a computer

Description automatically generated

Though memory usage didn’t improve, and CPU usage may have slightly increased, the runtime performance is significantly faster. This is consistent with the reduced loops and calculations that my optimisations brought.