Отчёт об ускорении работы программы

Изменение 1:

Добавление дополнительного условия в функции propagate_flow.

Код до:

```
tuple<Fixed<32, 16>, bool, pair<int, int>> propagate_flow(int x, int y, Fixed<32, 16> lim)
{
    last_use[x][y] = UT - 1;
    Fixed<32, 16> ret = 0.0;
    for (auto [dx, dy] : deltas) {
        int nx = x + dx, ny = y + dy;
        if (field[nx][ny] != '#' && last_use[nx][ny] < UT) {
            auto cap = velocity.get(x, y, dx, dy);
            auto flow = velocity_flow.get(x, y, dx, dy);
            if (flow == cap) {
                continue;
            }
            // assert(v >= velocity_flow.get(x, y, dx, dy));
            auto vp = min(lim, cap - flow);
            if (last_use[nx][ny] == UT - 1) {
                velocity_flow.add(x, y, dx, dy, vp);
                last_use[x][y] = UT;
                // cerr << x << " " << y << " -> " << nx << " " << ny << " " << vp << " / " << lim
<< "\n";
                return {vp, 1, {nx, ny}};
            }
            auto [t, prop, end] = propagate_flow(nx, ny, vp);
            ret += t;
            if (prop) {
                velocity_flow.add(x, y, dx, dy, t);
                last_use[x][y] = UT;
                // cerr << x << " " << y << " -> " << nx << " " << ny << " " << t << " / " << lim <<
"\n";
                return {t, prop && end != pair(x, y), end};
            }
        }
    }
    last_use[x][y] = UT;
    return {ret, 0, {0, 0}};
}
```

Код после:

```
tuple<Fixed<32, 16>, bool, pair<int, int>> propagate_flow(int x, int y, Fixed<32, 16> lim)
{
    last_use[x][y] = UT - 1;
    Fixed<32, 16> ret = 0.0;
    for (auto [dx, dy] : deltas) {
        int nx = x + dx, ny = y + dy;
        if (field[nx][ny] != '#' && last_use[nx][ny] < UT) {
            auto cap = velocity.get(x, y, dx, dy);
            auto flow = velocity_flow.get(x, y, dx, dy);
            if (flow == cap) {
                continue;
            }
            // assert(v >= velocity_flow.get(x, y, dx, dy));
            auto vp = min(lim, cap - flow);

            ////////////////////////////
            if (vp < 0.001)
                continue;
            ////////////////////////////
            if (last_use[nx][ny] == UT - 1) {
                velocity_flow.add(x, y, dx, dy, vp);
                last_use[x][y] = UT;
                // cerr << x << " " << y << " -> " << nx << " " << ny << " " << vp << " / " << lim
<< "\n";
                return {vp, 1, {nx, ny}};
            }
            auto [t, prop, end] = propagate_flow(nx, ny, vp);
            ret += t;
            if (prop) {
                velocity_flow.add(x, y, dx, dy, t);
                last_use[x][y] = UT;
                // cerr << x << " " << y << " -> " << nx << " " << ny << " " << t << " / " << lim <<
"\n";
                return {t, prop && end != pair(x, y), end};
            }
        }
    }
    last_use[x][y] = UT;
    return {ret, 0, {0, 0}};
}
```

Замеры времени работы на 1000 тиках:

До изменения:

- 20.7977 секунд;
- 21.3689 секунд;
- 21.1805 секунд;
- 21.5173 секунд;
- 21.3876 секунд.

После изменения:

- 0.90969 секунд;
- 1.12007 секунд;
- 0.977819 секунд;
- 0.911548 секунд;
- 0.909204 секунд.

Ускорение значительное.

Изменение 2:

Замена в структуре VectorField ranges::find на более быстрый switch-case.

Код до:

```
struct VectorField {
    Matrix<array<Fixed<32, 16>, deltas.size()>> v;
    Fixed<32, 16> &add(int x, int y, int dx, int dy, Fixed<32, 16> dv) {
        return get(x, y, dx, dy) += dv;
    }

    Fixed<32, 16> &get(int x, int y, int dx, int dy) {
        size_t i = ranges::find(deltas, pair(dx, dy)) - deltas.begin();
        assert(i < deltas.size());
        return v[x][y][i];
    }
};
```

Код после:

```
struct VectorField {
    Matrix<array<Fixed<32, 16>, deltas.size()>> v;
    Fixed<32, 16> &add(int x, int y, int dx, int dy, Fixed<32, 16> dv) {
```

```
        return get(x, y, dx, dy) += dv;
    }

    Fixed<32, 16> &get(int x, int y, int dx, int dy) {
        switch (4*dx + dy)
        {
            case -1: return v[x][y][2];
            case  1: return v[x][y][3];
            case -4: return v[x][y][0];
            case  4: return v[x][y][1];
        }
    }
};
```

Замеры времени работы на 1000 тиках:

До изменения:

- 20.7977 секунд;
- 21.3689 секунд;
- 21.1805 секунд;
- 21.5173 секунд;
- 21.3876 секунд.

После изменения:

- 19.9521 секунд;
- 19.8758 секунд;
- 19.852 секунд;
- 19.6002 секунд;
- 19.9314 секунд.

Ускорение небольшое, но присутствует.

Замеры времени работы на 1000 тиках после всех изменений:

До изменения:

- 20.7977 секунд;
- 21.3689 секунд;
- 21.1805 секунд;
- 21.5173 секунд;
- 21.3876 секунд.

После изменения:

- 0.864281 секунд;
- 0.956828 секунд;
- 0.931168 секунд;
- 0.880875 секунд;
- 0.889029 секунд.

Итог:

После всех изменений код стал работать в среднем на 96% быстрее.

Все измерения проводились с помощью инструментов из библиотеки std::chrono.