Projekt 1 – Hail cannon simulation

Mateusz Cyganek, Adrian Chrobot

# Uzyskany efekt

|  |  |
| --- | --- |
|  |  |
|  |  |

# Dodane stałe

W pliku pollution.hpp dodane zostały stałe, które są parametrami funkcji cannon() lub są wykorzystywane w zmodyfikowanych fragmentach funkcji compute\_rhs().

1. const int grid\_size = 40; // as defined in polution.cpp

2. const int cannon\_x\_loc = grid\_size / 2;

3. const int cannon\_shot\_time = 3’000;

4. const int cannon\_strength\_x = 15;

5. const int cannon\_strength\_y = 10;

6. const double cone\_limiter = 6.0;

7. const double wave\_speed = 2.0;

8. const double wave\_shortness = 3.0;

* grid\_size – rozmiar siatki, w której przeprowadzana jest symulacja,
* cannon\_x\_loc – lokalizacja działa na osi X, w symulacji działo znajduje się na środku siatki,
* cannon\_shot\_time – klatka, w której działo strzela,
* cannon\_strength\_x – mnożnik siły działa w osi X,
* cannon\_strength\_y – mnożnik siły działa w osi Y,
* cone\_limiter – zmienna ograniczenia maksymalnego kąta α rozchodzenia się wystrzelonej fali, równego π / cone\_limiter,
* wave\_speed – szybkość rozchodzenia się fali,
* wave\_shortness – szybkość zanikania fali.

# Funkcja Cannon

Poniższa tabela szczegółowo opisuje działanie funkcji.

|  |  |
| --- | --- |
| 1. double cannon(int x, int y, int iter) {  2. if (iter <= cannon\_shot\_time)  3. return 0.0;  4.  5. double i\_denom = (iterations / wave\_speed)  6. - cannon\_shot\_time;  7. // iteration denominator  8.  9. if (i\_denom <= 0)  10. return 0.0;  11.  12. double time = (iter - cannon\_shot\_time)  13. \* grid\_size / i\_denom;  14.  15. if (y > time)  16. return 0.0;  17.  18. int x\_prim = std::abs(cannon\_x\_loc - x);  19. double alpha\_rad = std::atan(x\_prim / time);  20.  21. if (alpha\_rad >= pi / cone\_limiter)  22. return 0.0;  23.  24. double y\_prim = std::sqrt(  25. time \* time - x\_prim \* x\_prim);  26.  27. if (y > y\_prim) // if y is higher than y'  28. return 0.0;  29.  30. return  31. wave\_shortness \* (y\_prim - y)  32. \* std::cos(alpha\_rad \* cone\_limiter \* 0.5);  33. } | • w przypadku gdy obecna iteracja iter jest przed wystrzałem zwracamy 0,      • obliczamy w którym momencie wystrzału jesteśmy i możymy przez rozmiar siatki by traktować time jako współrzędną osi Y, • jeżeli y > time znaczy to tyle, że y jest nad falą wystrzału • odległość x od działka   • ograniczenie kąta α rozchodzenia się fali  • y’ to przeciwprostokątna trójkąta o bokach x’, time, • jeżeli y > y’ to y jest nad falą wystrzału,  • różnica y i y’ zmniejsza wartość zwracaną wartość proporcjonalnie do odległości od fali, • cosinus kąta α zmniejsza zwracaną wartość na brzegach fali. |

# Modyfikacje funkcji compute\_rhs()

Plik polution.cpp nie został zmieniony

Funkcja compute\_rhs() znajduje się w pliku polution.hpp. Zostały dodane do niej zaznaczone fragmenty.

1. for (auto a : dofs\_on\_element(e)) {

2. value\_type v = eval\_basis(e, q, a);

3. value\_type u = eval\_fun(u\_prev, e, q);

4.

5. double b = cannon(e[0], e[1], iter);

6. double bx = (cannon(e[0] - 1, e[1], iter) - b) \* cannon\_strength\_x;

7. double by = (cannon(e[0], e[1] - 1, iter) - b) \* cannon\_strength\_y;

8. double gradient\_prod = k\_x \* u.dx \* v.dx + k\_y \* u.dy \* v.dy;

9. double h = e2h(e[1]);

10. double val =

11. u.val \* v.val

12. - steps.dt \* gradient\_prod

13. + steps.dt \* dTy(h) \* u.dy \* v.val

14. + steps.dt \* f(h) \* v.val

15. - steps.dt \* bx \* u.dx \* v.val

16. - steps.dt \* by \* u.dy \* v.val;

17.

18. rhs(a[0], a[1]) += val \* w \* J;

19. }

* b – wartość funkcji cannon() w punkcie, w danej iteracji iter,
* bx – składowa x wektora pola fali wystrzału w punkcie w danej iteracji iter, uzyskana z różnicy zwracanych wartości funkcji cannon() w punkcie (x, y) oraz (x-1, y)
* by – składowa y wektora pola fali wystrzału w punkcie w danej iteracji iter, uzyskana z różnicy zwracanych wartości funkcji cannon() w punkcie (x, y) oraz (x, y - 1)

Następnie bx, by są mnożone przez stałe siły działka - cannon\_strength\_x, cannon\_strength\_y, co umożliwia kontrole jego mocy, i odejmowane od val, w linijce 15 oraz 16, celem uzyskania wartości w punkcie (x, y) pomniejszonej o efekt fali wystrzału.