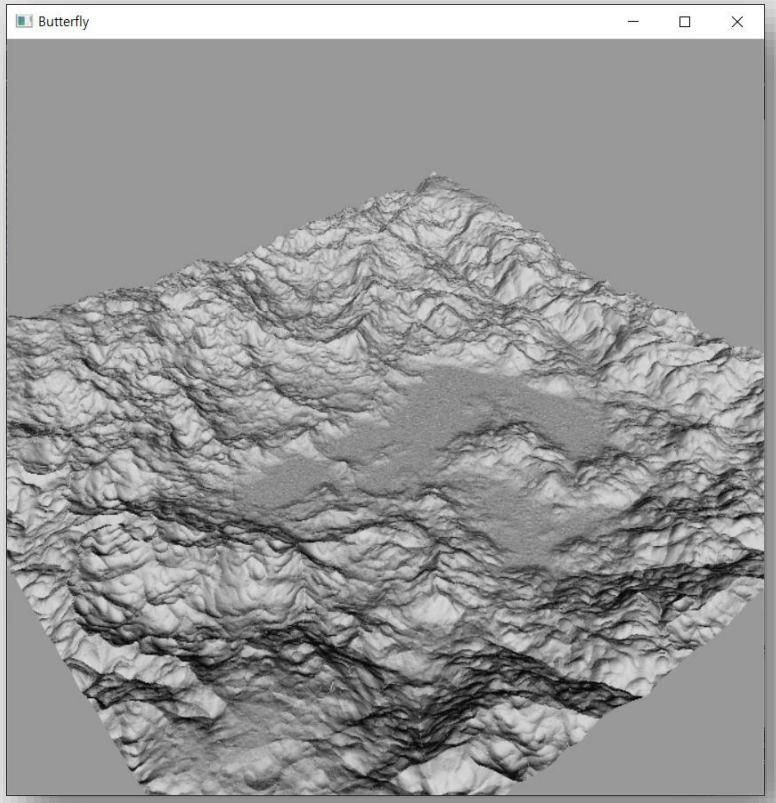
Chainmail algorithm
With simple voxelization

컴퓨터그래픽스응용
소프트웨어응용학부
201704060
안장훈

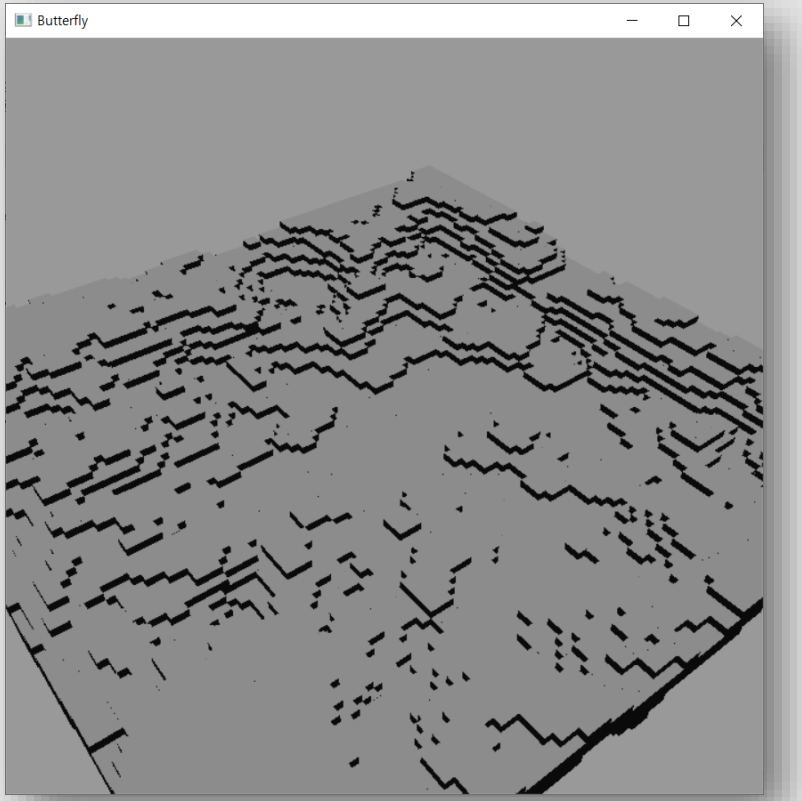
목차

Voxelization



Before
Vertex : 1002001

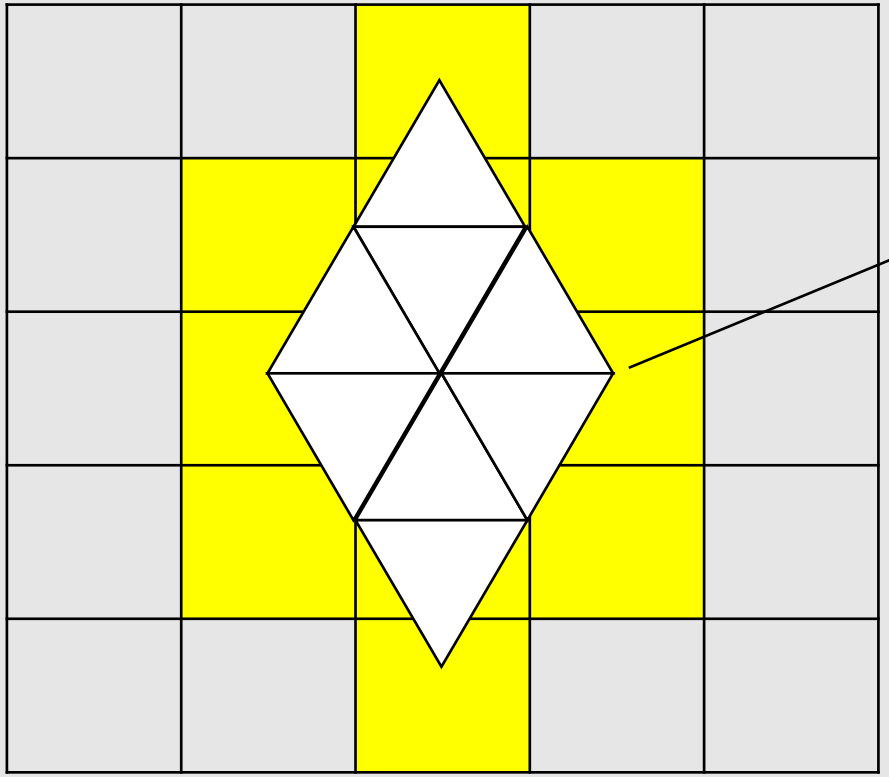
Voxelization
256x256

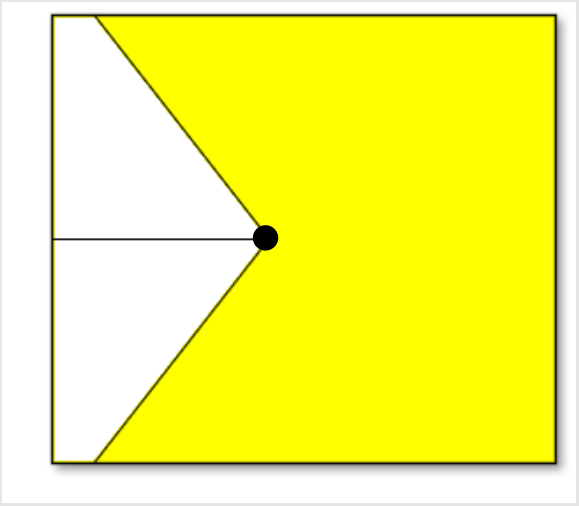


After
Vertex(Voxel) : 12792

Voxelization

algorithm



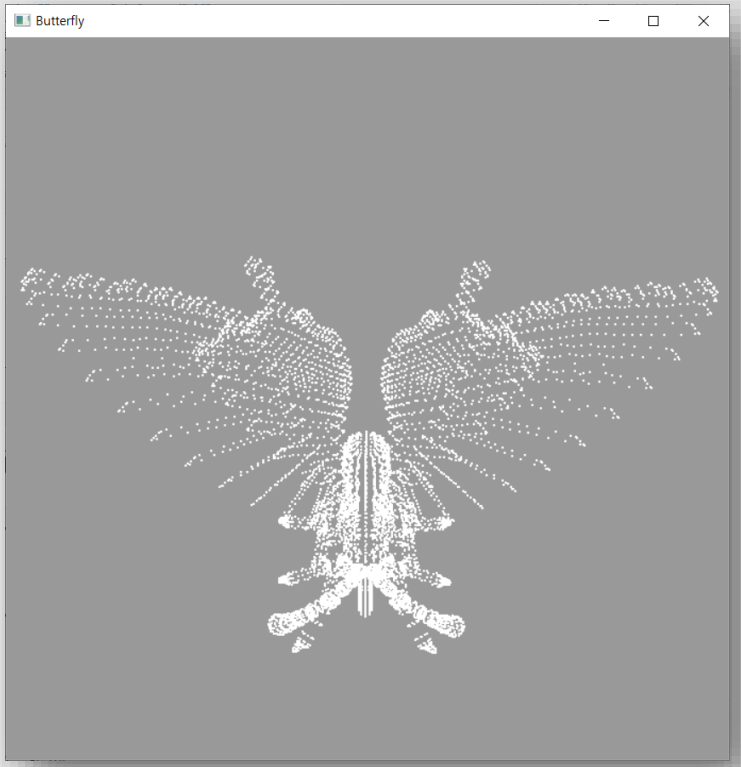


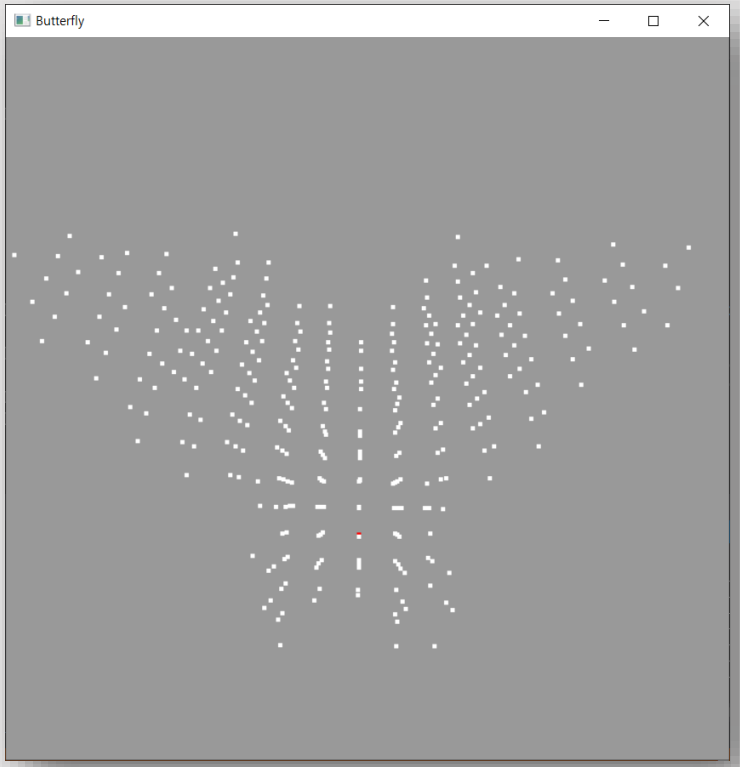
```
double grid_coord_calc(double n, double size, double max, double min) {
    size += -1;
    double p = n * 2 - (size - 1);// -size + size
    p /= (size - 1); //-1 ~ 1
    p *= (max + abs(min)) / 2; //max , min noramlize range
    p += (max - abs(min)) / 2; //-min ~ max
    return p;
}
```

3D mesh와 Voxel grid

Voxelization

algorithm





3D mesh vertex

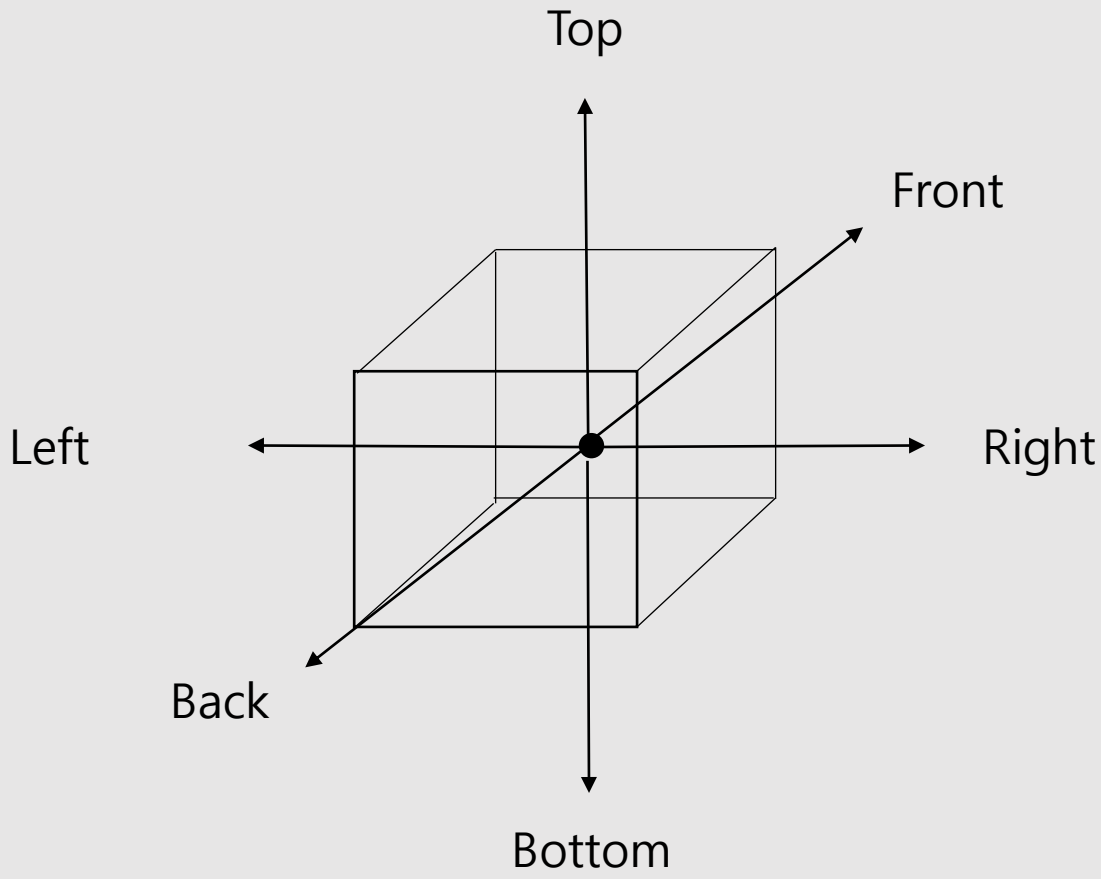
3D voxel vertex

Voxelization

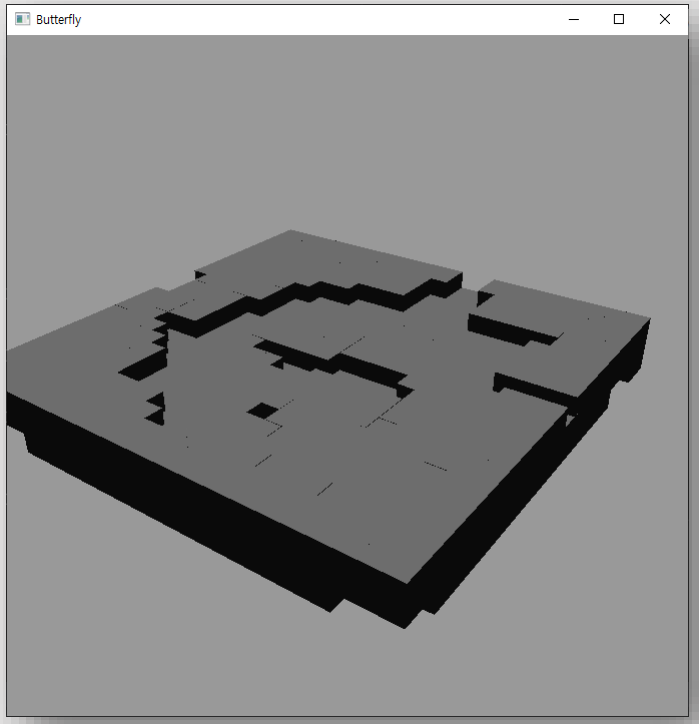
Build adjacency

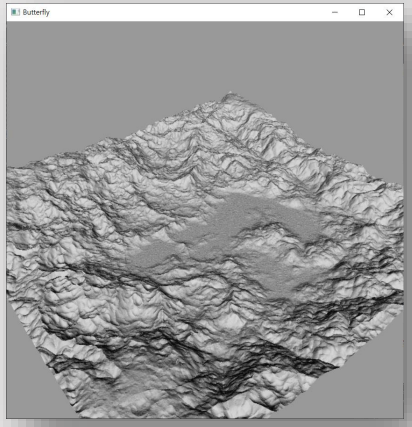
class Voxel

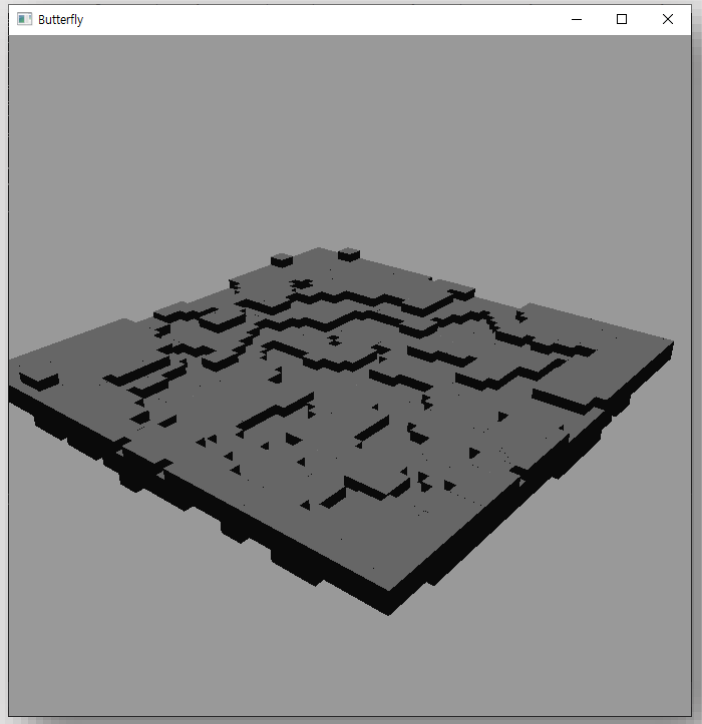
```
public: // Neighbor voxels
    Voxel*              left = nullptr;
    Voxel*              right = nullptr;
    Voxel*              top = nullptr;
    Voxel*              bottom = nullptr;
    Voxel*              front = nullptr;
    Voxel*              back = nullptr;
```



Voxelization



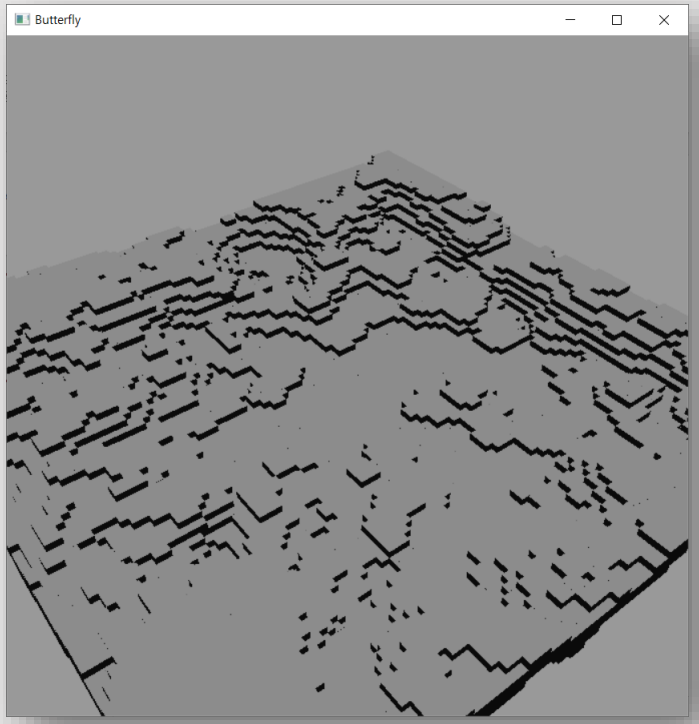


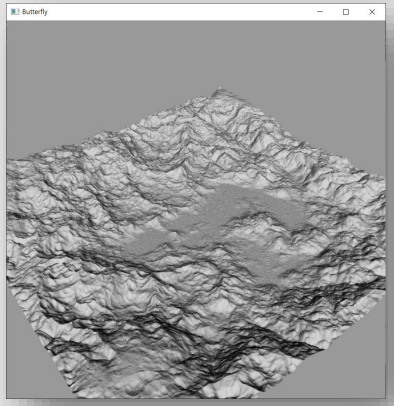


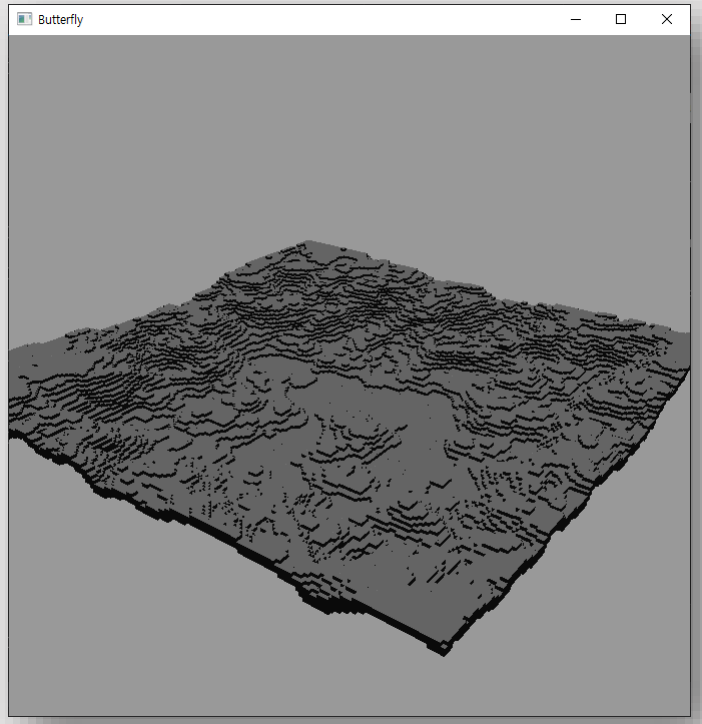
Size : 64

Size : 128

Voxelization







Size : 256

Size : 512

Chainmail algorithm

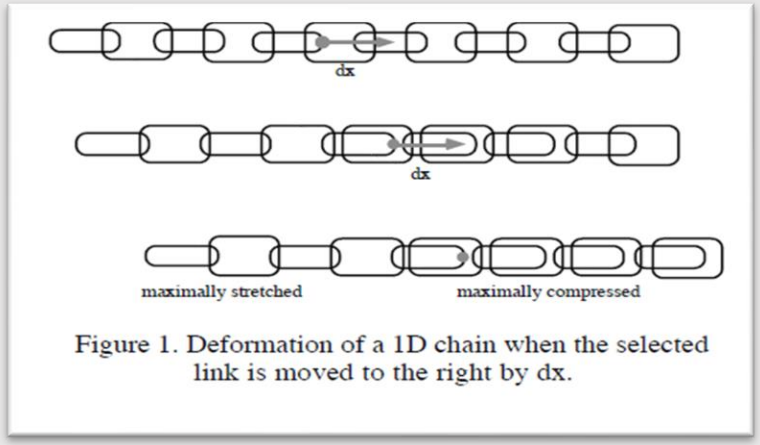


Figure 1. Deformation of a 1D chain when the selected link is moved to the right by dx.

1D

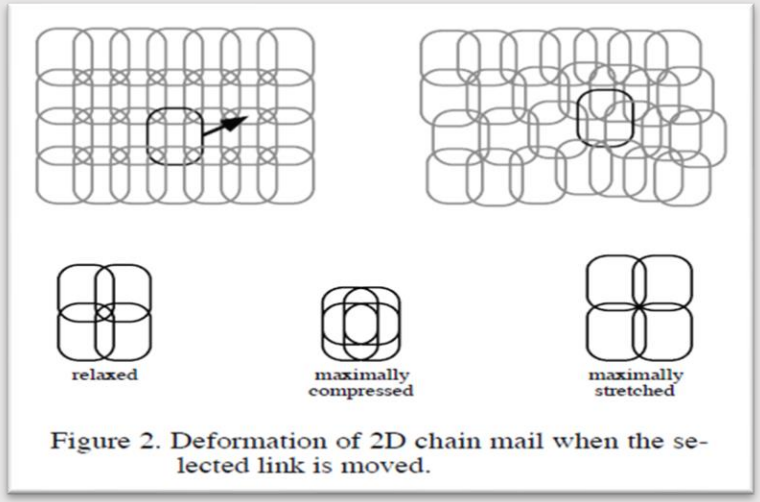


Figure 2. Deformation of 2D chain mail when the selected link is moved.

2D

Looks like chainmail!

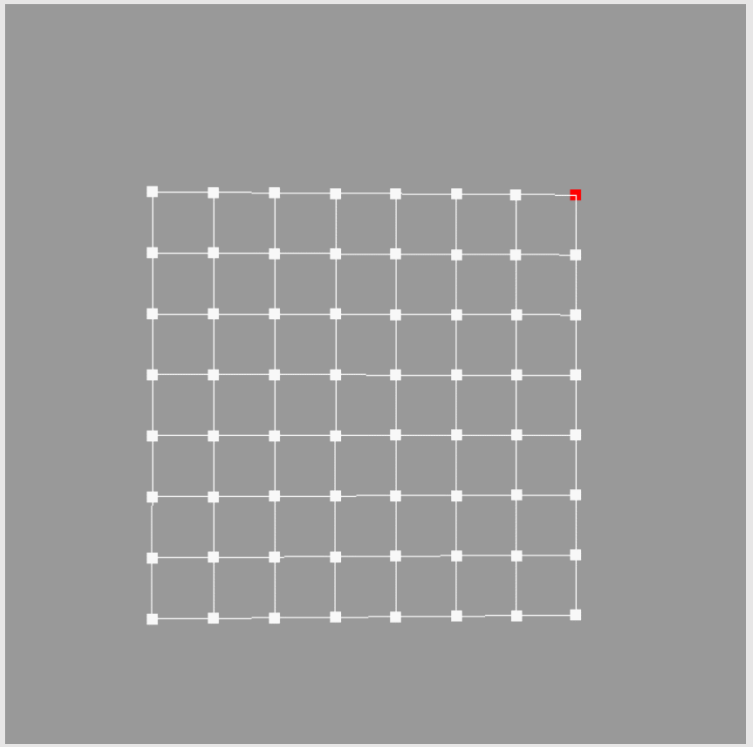
S.F. Gibson, "3D Chainmail: A Fast Algorithm for Deforming Volumetric Objects," Proceeding of Symposium on Interactive 3D Graphics, 1997

Chainmail algorithm

propagate

$$
\begin{aligned}
&\text{if } (x - x_{left}) < minDx,\ x = x_{left} + minDx;\\
&\text{else if } (x - x_{left}) > maxDx,\ x = x_{left} + maxDx;\\
&\\
&\text{if } (y - y_{left}) < -maxHorixDy,\ y = y_{left} - maxHorixDy;\\
&\text{else if } (y - y_{left}) > maxHorixDy,\ y = y_{left} + maxHorixDy;
\end{aligned}
$$

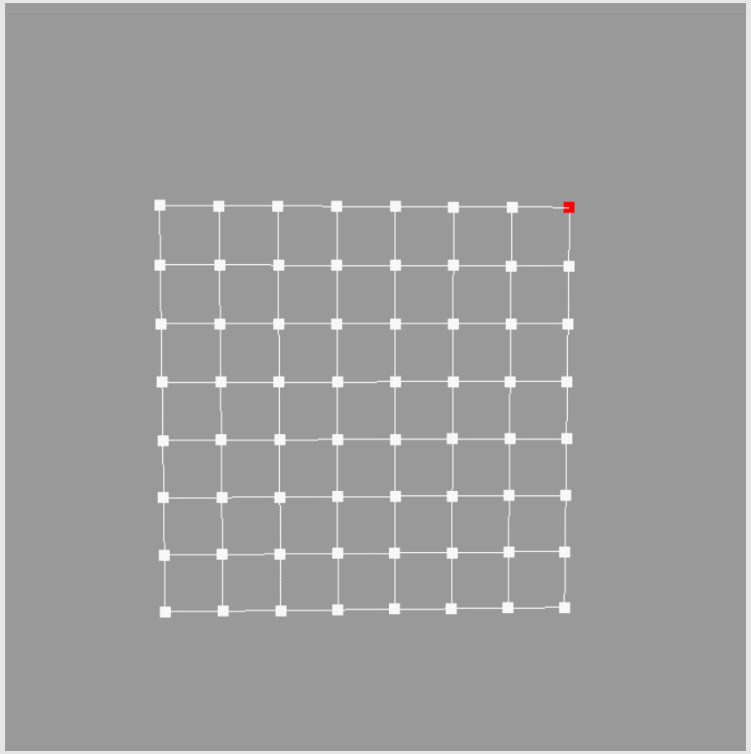
Right direction



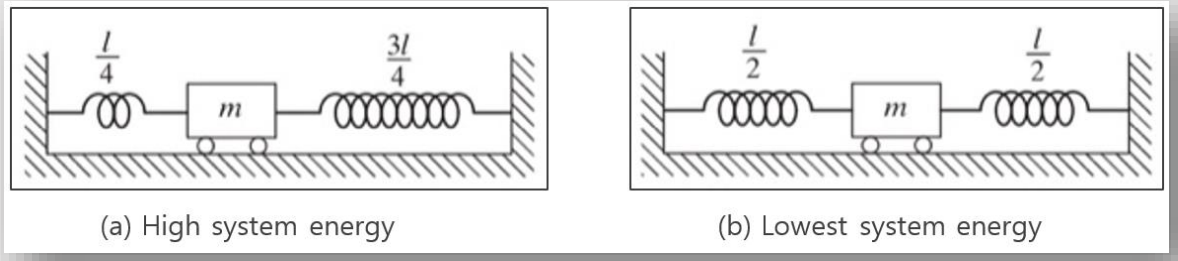
2D Chainmail

Chainmail algorithm

Elastic relaxation



2D Chainmail



(a) High system energy (b) Lowest system energy

$$(x,y,z)_{opt} = \left(\frac{1}{N}\sum_{neighbors}(x_n - \Delta x_n)_*, \quad \frac{1}{N}\sum_{neighbors}(y_n - \Delta y_n)_* \right),$$

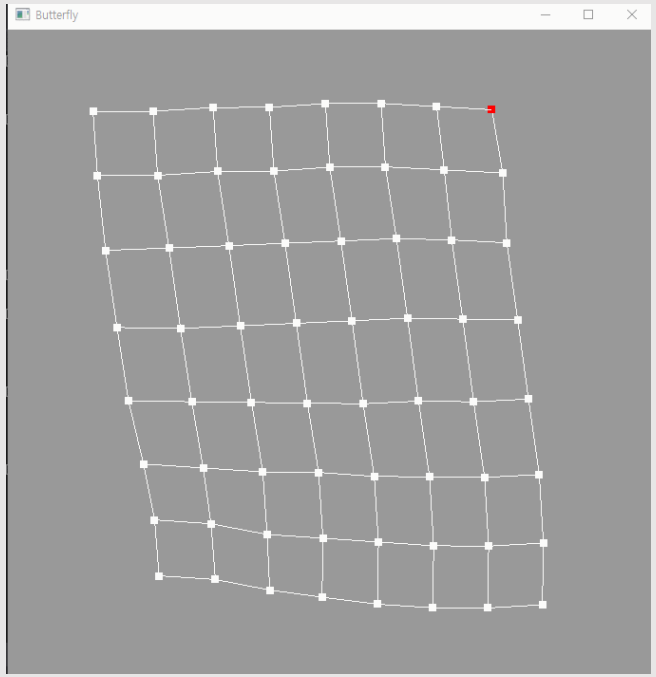
$$\Delta x_n = \begin{cases} -\Delta x & : \ n = left \\ +\Delta x & : \ n = right \\ 0 & : \ \text{all other neighbors} \end{cases}$$

$$\Delta y_n = \begin{cases} -\Delta y & : \ n = bottom \\ +\Delta y & : \ n = top \\ 0 & : \ \text{all other neighbors} \end{cases}$$

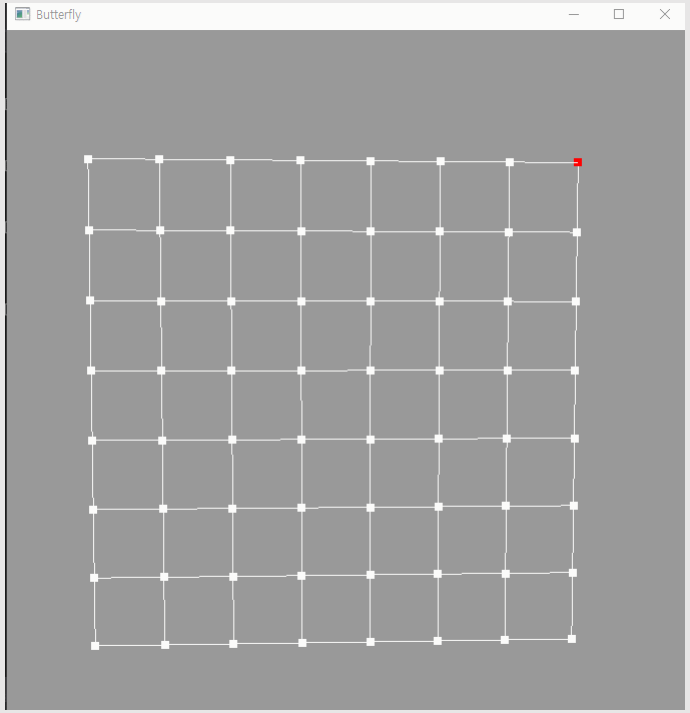
Relaxing algorithm

Chainmail algorithm

Elastic relaxation



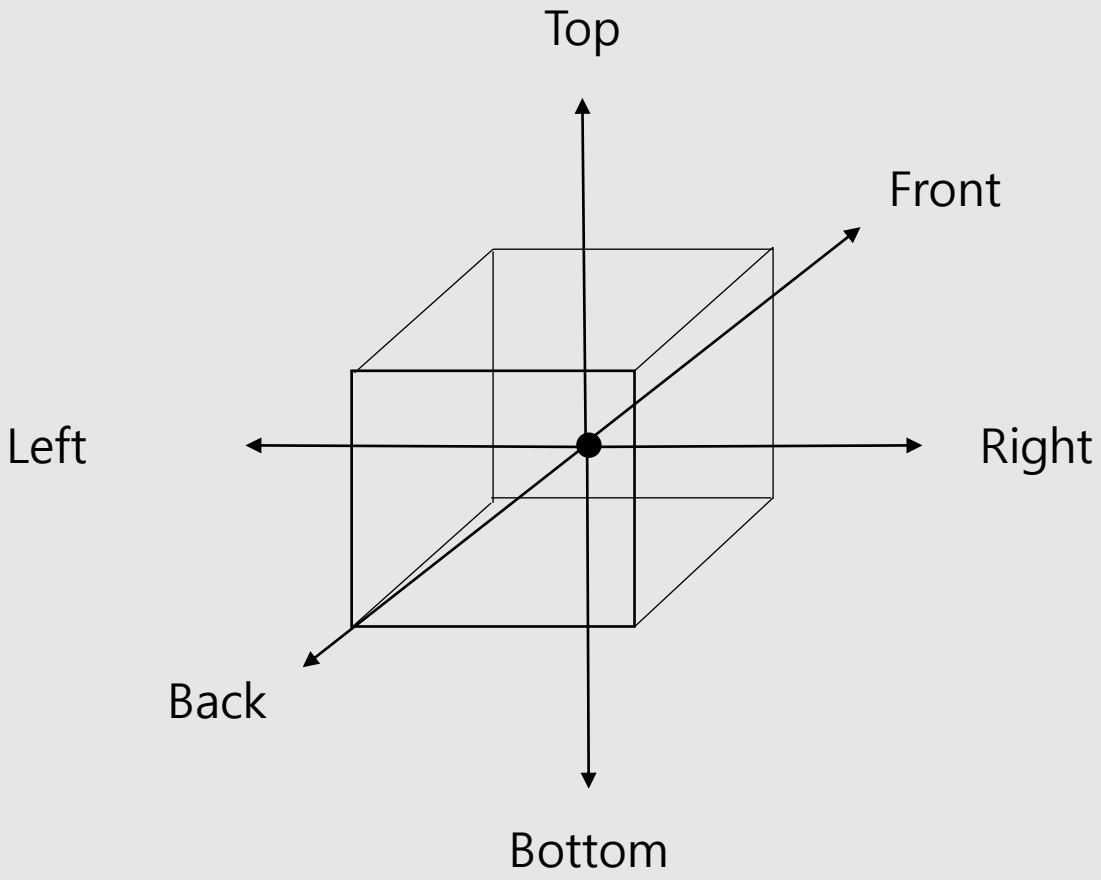
k = 0.1

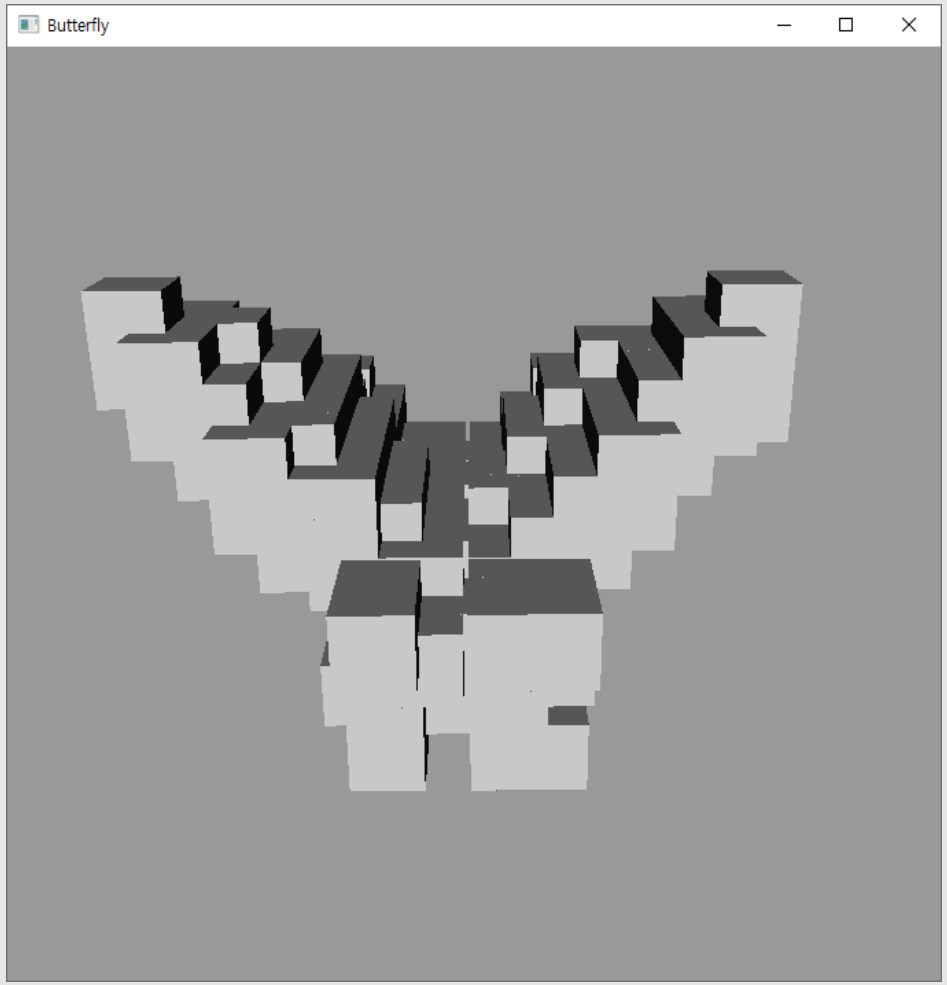


k = 0.005

Chainmail algorithm

3D Voxel Structure

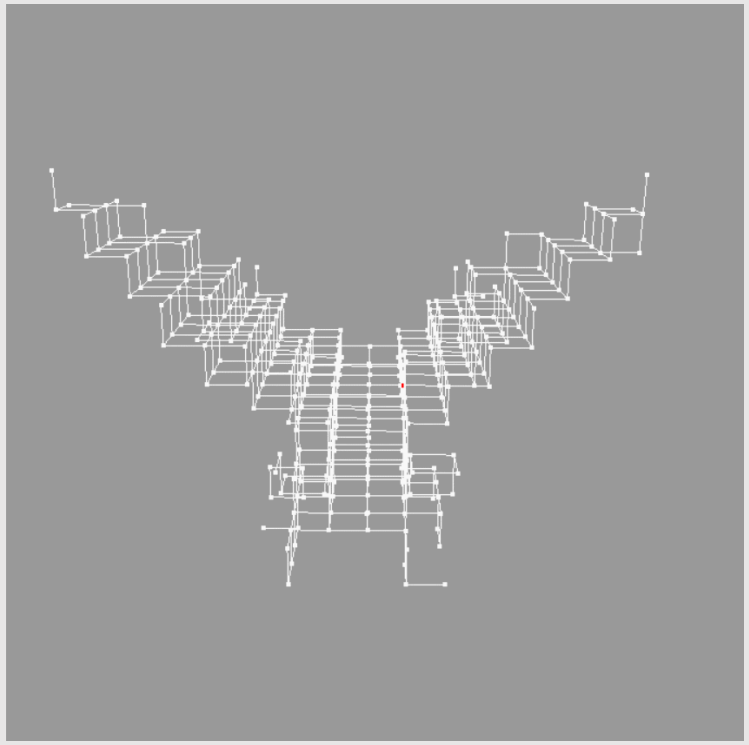




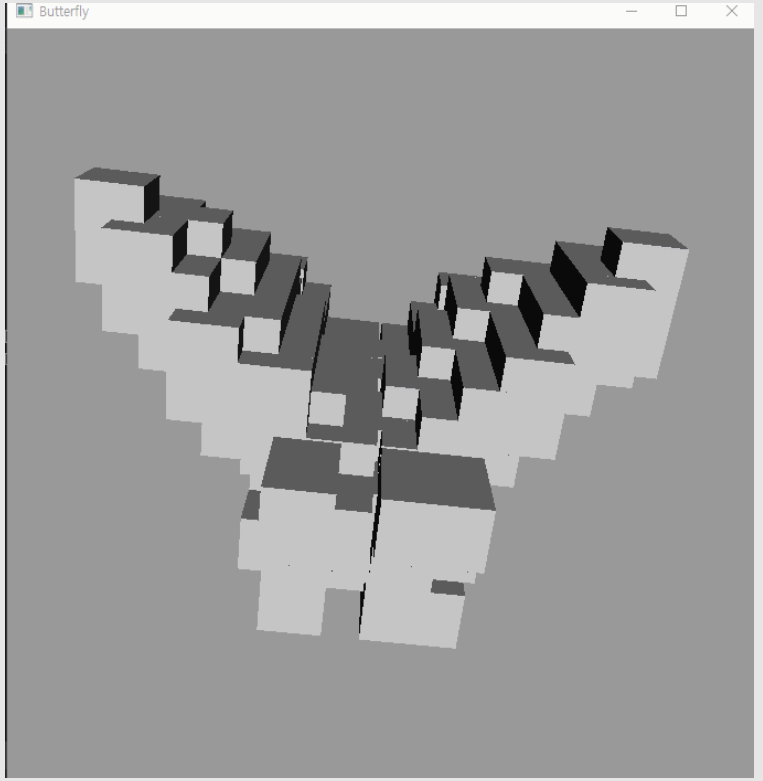
Build adjacency

Chainmail algorithm

3D Voxel Structure



Draw wire



Draw voxel

Butterfly - Microsoft Visual Studio

파일(F) 편집(E) 보기(V) 프로젝트(P) 빌드(B) 디버그(D) 팀(M) Nsight 도구(T) 테스트(S) 분석(N) 창(W) 도움말(H)

Release x64 로컬 Windows 디버거

Mesh.cpp | Physics.cpp | main.cpp | Voxel.h | Vertex.cpp | Calc.cpp | Vertex.h | Face.h | IBMCell.cpp | IBM.cpp | Face.cpp | Physics.h | Voxel.cpp | IBMCell.h

Butterfly (전역 범위) main(int argc, char ** argv)

```
185     else if (buttons[0]) {
186         rot_x += (float)0.2f * diff_y;
187         rot_y += (float)0.2f *diff_x;
188     }
189
190     glutPostRedisplay();
191 }
192
193 void main(int argc, char** argv) {
194     int chose = 0;
195     cout << "Chose obj number (0 : Butterfly, 1 : Terrain, 2 : Quad) >> "; cin >> chose;
196     if (chose == 1) {
197         mesh = new Mesh("OBJ₩₩Terrain.obj", 256);
198     }
199     else if(chose == 0){
200         mesh = new Mesh("OBJ₩₩Butterfly.obj", 16);
201         physics.set_propertys(0.7, 0.3);
202     }
203     else if (chose == 2) {
204         mesh = new Mesh("OBJ₩₩Quad.obj", 8); // No faces!
205         physics.set_propertys(0.9, 0.3);
206     }
207
208
209     glutInit(&argc, argv);
210     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
211     glutInitWindowSize(680, 680);
212     glutInitWindowPosition(100, 100);
213
214     glutCreateWindow("Butterfly");
215     glutDisplayFunc(Display);
216     glutReshapeFunc(Reshape);
217     glutMouseFunc(Mouse);
```

128 %

출력

출력 보기 선택(S): 디버그

```
'Butterfly.exe'(Win32): 'C:₩Windows₩System32₩imm32.dll'을(를) 로드했습니다. PDB 파일을 찾거나 열 수 없습니다.
0x2c24 스레드가 종료되었습니다(코드: 0 (0x0)).
0x464c 스레드가 종료되었습니다(코드: -1073741510 (0xc000013a)).
0x2f4c 스레드가 종료되었습니다(코드: -1073741510 (0xc000013a)).
0x38b0 스레드가 종료되었습니다(코드: -1073741510 (0xc000013a)).
'[0464] Butterfly.exe' 프로그램이 종료되었습니다(코드: -1073741510 (0xc000013a)).
```

오류 목록 기호 찾기 결과 출력

솔루션 탐색기

솔루션 탐색기 검색(Ctrl+;)

솔루션 'Butterfly' (1개 프로젝트)

- Butterfly
  - 참조
  - 외부 종속성
  - 리소스 파일
  - 소스 파일
    - Calc.cpp
    - Face.cpp
    - IBM.cpp
    - IBMCell.cpp
    - main.cpp
    - Mesh.cpp
    - Physics.cpp
    - Vertex.cpp
    - Voxel.cpp
  - 헤더 파일
    - Calc.h
    - Face.h
    - IBM.h
    - IBMCell.h
    - Mesh.h
    - Physics.h
    - Vec3.h
    - Vertex.h
    - Voxel.h

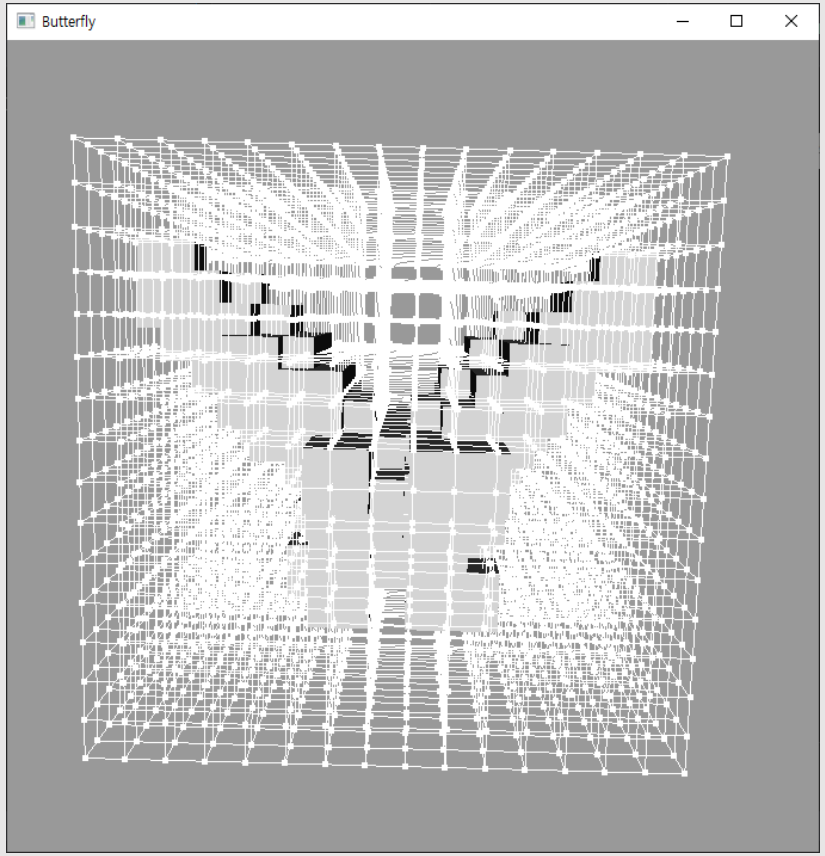
속성

준비 줄: 204 열: 58 문자: 52 INS 소스 제어에 추가

오전 1:23 2021-12-15

아쉬운 점

Immersed boundary method

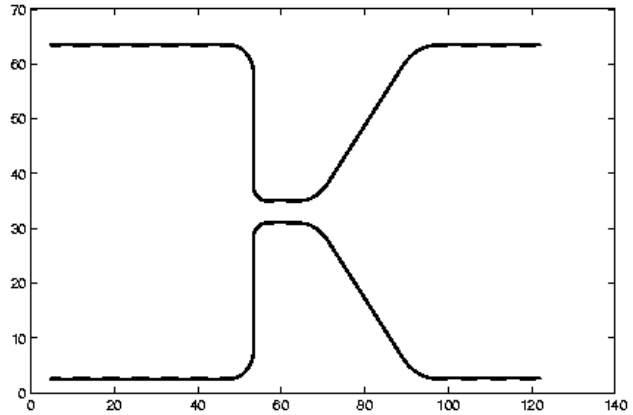


IBM grid coordination

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_j u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j \partial x_j} + f_i$$

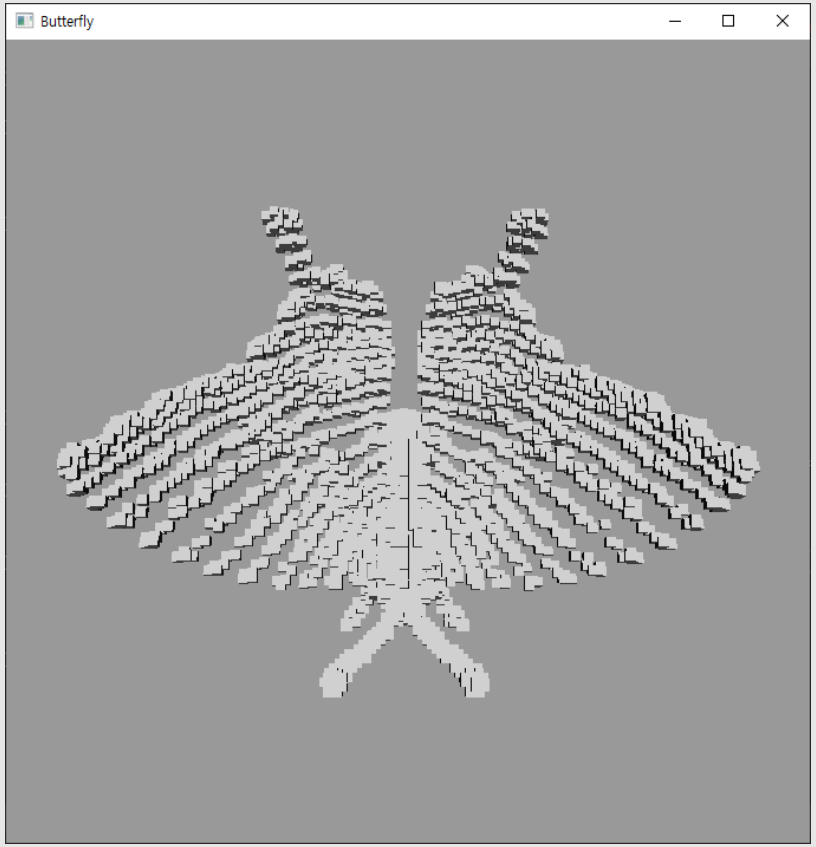
$$\frac{\partial u_i}{\partial x_i} - q = 0$$

IBM governing equation

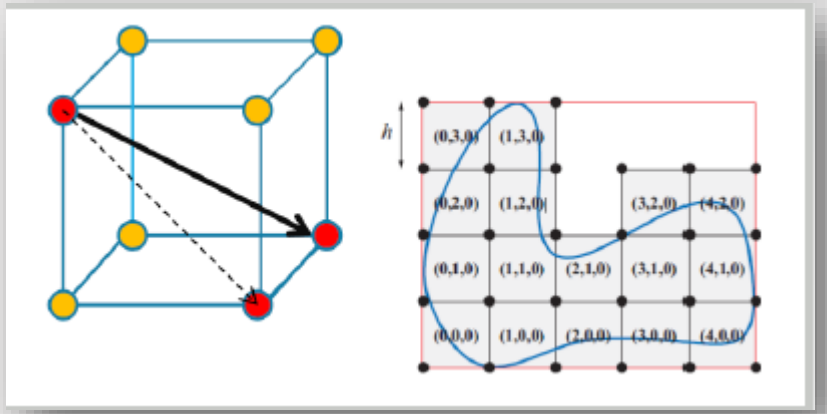


Vocal fold simulation

아쉬운 점



Size : 128



감사합니다!

컴퓨터그래픽스응용
소프트웨어응용학부
201704060
안장훈