Interective Visualization PA#2

Mass-spring-simulation report

2016320189 이광렬

목차

1. 구현 목표

1) Mass를 갖는 Node를 Spring으로 연결하여 옷감 모델 제작.

2) 수치 적분 알고리즘을 이용하여 옷감의 움직임 구현.

3) 3차원 렌더링을 적용하여 옷감의 입체감 표현.

2. 구현 과정

A. 기본 구현 사항

i. Initialization

1. Node

① Simulator.cpp에서 설정한 cloth값을 바탕으로 노드 생성

② 노드의 생성시 시뮬레이션을 위해 y, z 좌표 change

```
for (int x = size_x; x > 0; x--) {
    for (int y = size_y; y > 0; y--) {
        int z = size_z;
        Node *xp = new Node(vec3(size_x-x-25, z, size_y-y-25));
        xp->inipos = vec3(size_x - x, z, size_y - y)/49;
        nodes.push_back(xp);
    }
}
```

cloth.h - void mass_cloth::init()

③ 시뮬레이션을 위해 (x,y,z) = (0,0,0),(0,0,50) 노드 2개 fix

```
nodes[0]->isFixed = true;
nodes[49]->isFixed = true;
```

cloth.h - void mass_cloth::init()

2. Structural spring

① 생성한 Node 사이 Structural spring 연결

```
for (int x = 0; x < size_x; x++) {
    for (int y = 0; y < size_y-1; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50*x+y], nodes[50*x+y+1]);
        sp->spring_coef = structural_coef;
        spring.push_back(sp);
    }
}
for (int x = 0; x < size_x - 1; x++) {
    for (int y = 0; y < size_y; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50 * x + y], nodes[50 * (x + 1) + y]);
        sp->spring_coef = structural_coef;
        spring.push_back(sp);
    }
}
```

cloth.h - void mass_cloth::init()

3. Shear Spring

① 생성한 Node 사이 Shear spring 연결

```
for (int x = 0; x < size_x - 1; x++) {
    for (int y = 0; y < size_y - 1; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50 * x + y], nodes[50 * (x + 1) + y + 1]);
        sp->spring_coef = shear_coef;
        spring.push_back(sp);
    }
}
for (int x = 0; x < size_x - 1; x++) {
    for (int y = 1; y < size_y; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50 * x + y], nodes[50 * (x + 1) + y - 1]);
        sp->spring_coef = shear_coef;
        spring.push_back(sp);
    }
}
```

cloth.h - void mass_cloth::init()

4. Face

① 생성한 Node를 묶어 face 생성 (3개씩 묶어 face를 생성)

```
for (int x = 0; x < size_x-1; x++) {
    for (int y = 0; y < size_y - 1; y++) {
        faces.push_back(nodes[50 * x + y]);
        faces.push_back(nodes[50 * (x + 1) + y + 1]);
        faces.push_back(nodes[50 * (x + 1) + y]);

        faces.push_back(nodes[50 * x + y]);
        faces.push_back(nodes[50 * x + y + 1]);
        faces.push_back(nodes[50 * (x + 1) + y + 1]);
    }
}
```

cloth.h - void mass_cloth::init()

② 생성한 face를 그림 (Rendering에 부가 설명)

ii. Simulation

1. Internal Force 계산

① Spring Force + Damping Force 계산(Hooke's Law +Damping Force 사용)

$$\mathbf{f} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}\left[k_s\left(\left|\mathbf{x}_j - \mathbf{x}_i\right| - l_o\right) + k_d(\mathbf{v}_j - \mathbf{v}_i)\cdot\frac{\mathbf{x}_j - \mathbf{x}_i}{\left|\mathbf{x}_j - \mathbf{x}_i\right|}\right]$$

Hooke's Law + Damping Force

```
void internal_force(double dt)
{
    vec3 normalizd = (p1->position - p2->position).Normalizevec();
    vec3 force = (normalizd) * (spring_coef*(p1->position.dist(p2->position) - initial_length) +
damping_coef * (p1->velocity - p2->velocity).dot(normalizd));
    p1->add_force(-1*force);
    p2->add_force(force);
}
```

spring.h – void mass_spring::internal_force(double dt)

2. Integration (Euler's Explicit Method 사용)

```
vector3d x_new, v_new, a;
a = force/mass;
v_new = v + a*dt;
x_new = x+ v*dt;
```

Euler's Explicit Method

```
void integrate(double dt)
{
    if (!isFixed)
    {
        if (position.y < -14.5) {
            position.y = -14.5;
            velocity.y = 0;
        }
        acceleration = force / mass;
        velocity = acceleration * dt + velocity;
        position = position + velocity * dt;
```

Node.h – void Node::integrate(double dt)

① 바닥에 옷감이 오래 닿아있으면 바닥을 뚫는 현상 해결

② Euler's Explicit Method 적용

iii. Rendering

1. 각각 노드(Vertex)에서의 Normal값을 계산하여 lighting 적용

① 각 node의 normal을 초기화

② 각 face의 normal을 계산

③ 인접한 node의 normal에 face의 normal을 더함 (Node에서 인접한 face의 normal을 더하는 것과 같음)

④ 각 node의 normal을 nomalize

```
void computeNormal()
{
    for (int i = 0; i < nodes.size(); i++) {
        nodes[i]->normal = vec3(0,0,0);
    }
    for (int i = 0; i < faces.size(); i += 3) {
        vec3 p1 = faces[i + 1]->position - faces[i]->position;
        vec3 p2 = faces[i + 2]->position - faces[i]->position;
        vec3 N = p1.Cross(p2);
        vec3 NN = N.Normalizevec();
        faces[i]->normal += NN;
        faces[i + 1]->normal += NN;
        faces[i + 2]->normal += NN;
    }
    for (int i = 0; i < nodes.size(); i++) {
        nodes[i]->normal.Normalize();
    }
}
```

cloth.h – void mass_cloth::computeNormal()

```
case DRAW_FACES:
    glColor3f(0.5f, 0.5f, 0.5f);
    glEnable(GL_LIGHTING);
    glEnable(GL_TEXTURE_2D);
    for (int i = 0; i < faces.size(); i = i + 3) {
        glBegin(GL_TRIANGLES);
            glTexCoord2f(faces[i]->inipos.x,faces[i]->inipos.z);
            glNormal3f(faces[i]->normal.x, faces[i]->normal.y, faces[i]->normal.z);
            glVertex3f(faces[i]->getPosX(),faces[i]->getPosY(),faces[i]->getPosZ());
            glTexCoord2f(faces[i+1]->inipos.x, faces[i+1]->inipos.z);
            glNormal3f(faces[i+1]->normal.x, faces[i+1]->normal.y, faces[i+1]->normal.z);
            glVertex3f(faces[i + 1]->getPosX(), faces[i + 1]->getPosY(), faces[i + 1]->getPosZ());
            glTexCoord2f(faces[i+2]->inipos.x, faces[i+2]->inipos.z);
            glNormal3f(faces[i+2]->normal.x, faces[i+2]->normal.y, faces[i+2]->normal.z);
            glVertex3f(faces[i + 2]->getPosX(), faces[i + 2]->getPosY(), faces[i + 2]->getPosZ());
        glEnd();
    }
    glDisable(GL_TEXTURE_2D);
```

Face modeling code (각 face의 node에서 normal, position 값을 읽어옴)

Draw.cpp – void mass_cloth::draw()

iv. Collision

1. 바닥면과 cloth의 collision detection, response를 모델링하여 적용

① 바닥면과 cloth의 collision detection

i. 바닥면의 법선 벡터(0,1,0)과 계산

$$\text{If } ((x-p)\cdot n < r \;\&\&\; (n\cdot v < 0))$$

② Response를 적용

```
void collision_response(vec3 ground)
{
    for (int i = 0; i < nodes.size(); i++) {
        if ((nodes[i]->position - ground).dot(vec3(0, 1, 0)) <= 1 && vec3(0, 1, 0).dot(nodes[i]->velocity) < 0) {
            nodes[i]->velocity.y *= -0.5;
        }
    }
}
```

cloth.h – coid collision_reponse(vec3 ground)

v. User Interaction

1. 마우스 드래그 방향에 따른 외력 적용

① Mouse 입력 정도에 해당하는 힘을 계산

② 시뮬레이션에서 회전한 만큼 힘을 회전

③ 계산한 힘을 cloth에 적용

```
else if (interationMode)
{
    GLfloat matrix[16];
    glPushMatrix();
    glLoadIdentity();
    glRotatef(m_Rotate[0], 1.0, 0.0, 0.0);
    glRotatef(m_Rotate[1], 0.0, 1.0, 0.0);
    glGetFloatv(GL_MODELVIEW_MATRIX,matrix);
    glPopMatrix();
    vec3 userforce = vec3(diffx,diffy,0);
    userforce.x = (matrix[0] * userforce.x + matrix[4] * userforce.y + matrix[8] * userforce.z + matrix[12] * 1);
    userforce.y = -(matrix[1] * userforce.x + matrix[5] * userforce.y + matrix[9] * userforce.z + matrix[13] * 1);
    userforce.z = matrix[2] * userforce.x + matrix[6] * userforce.y + matrix[10] * userforce.z + matrix[14] * 1;
    S_Simulator.cloth->add_force(userforce);
}
```

Viewer.cpp – void Viewer::Motion(int x, int y)

B. 추가 구현 사항

i. Bending spring

1. Bending Spring을 추가하여 Bending 효과 적용

① Node 사이 bending spring 연결

```
for (int x = 0; x < size_x; x++) {
    for (int y = 0; y < size_y - 2; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50 * x + y], nodes[50 * x + y + 2]);
        sp->spring_coef = bending_coef;
        spring.push_back(sp);
    }
}
for (int x = 0; x < size_x - 2; x++) {
    for (int y = 0; y < size_y; y++) {
        int z = size_z;
        mass_spring *sp = new mass_spring(nodes[50 * x + y], nodes[50 * (x + 2) + y]);
        sp->spring_coef = bending_coef;
        spring.push_back(sp);
    }
}
```

cloth.h - void mass_cloth::init()

ii. Sphere와의 충돌처리

1. Sphere를 생성하고 옷감과의 collision-response 적용

① Sphere 생성

```
vec3                    sphere;
float                   radius;
bool                    m_Sphere;
```

Sphere 정의

Simulator.h – class Simulator

```
sphere = vec3(0, -5, 0);
radius = 5.0f;
```

Sphere 위치, 크기 선언

Simulator.cpp – void Simulator::Initialize()

```
case '4':
    S_Simulator.m_Sphere = !S_Simulator.m_Sphere;
    break;
```

Keyboard 4 입력시 생성 / 제거

Viewer.cpp – void Viewer::Keyboard(unsigned char key, int x, int y)

```
if(m_Sphere)    DrawSphere();
```

Simulator.cpp – void Simulator::Render()

```
void Simulator::DrawSphere() {
    glColor3f(0.0f, 1.0f, 1.0f);
    GLUquadric *sp = gluNewQuadric();
    glPushMatrix();
    glTranslatef(sphere.x, sphere.y, sphere.z);
    gluSphere(sp, radius, 50, 20);
    glPopMatrix();
}
```

Sphere Draw

Simulator.cpp – void Simulator::DrawSphere()

② Collision-response 적용

i. Collision detection

가) 구의 중심 – 노드 간의 벡터와 연산 (평면에서의 법선 벡터)

ii. Collision-Response

가) 각 노드가 구의 중심으로 향하는 힘을 변환

$$v_N = (N \cdot v)N$$
$$v_T = v - v_N$$
$$V' = V_T - V_N$$

iii. Texture Mapping

1. 원하는 Image나 Texture를 사용하여, 옷감에 Texture Mapping 적용

① 각 노드별 위치-Texture coordinates

```
xp->inipos = vec3(size_x - x, z, size_y - y)/49;
```

(Node 생성 코드의 일부)

cloth.h - void mass_cloth::init()

```
glTexCoord2f(faces[i]->inipos.x,faces[i]->inipos.z);
```

(Face rendering 코드의 일부)

Draw.cpp – void mass_cloth::draw()

② Image load

```
GLuint LoadTexture(const char * filename)
{

    GLuint texture;

    int width, height;

    unsigned char * data;

    FILE * file;

    file = fopen(filename, "rb");

    if (file == NULL) return 0;
    width = 900;
    height = 900;
    data = (unsigned char *)malloc(width * height * 3);
    fread(data, 54, 1, file);
```

fread(54bit):bmp image header(예외처리)

width,height : image size

```
fread(data, width * height * 3, 1, file);
fclose(file);

for (int i = 0; i < width * height; ++i)
{
    int index = i * 3;
    unsigned char B, R;
    B = data[index];
    R = data[index + 2];

    data[index] = R;
    data[index + 2] = B;

}
```

Bmp image : BGR 순서로 data가 있음. 이를 RGB순으로 재배치

```
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_NEAREST);


glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
gluBuild2DMipmaps(GL_TEXTURE_2D, 3, width, height, GL_RGB, GL_UNSIGNED_BYTE, data);
free(data);

return texture;
```

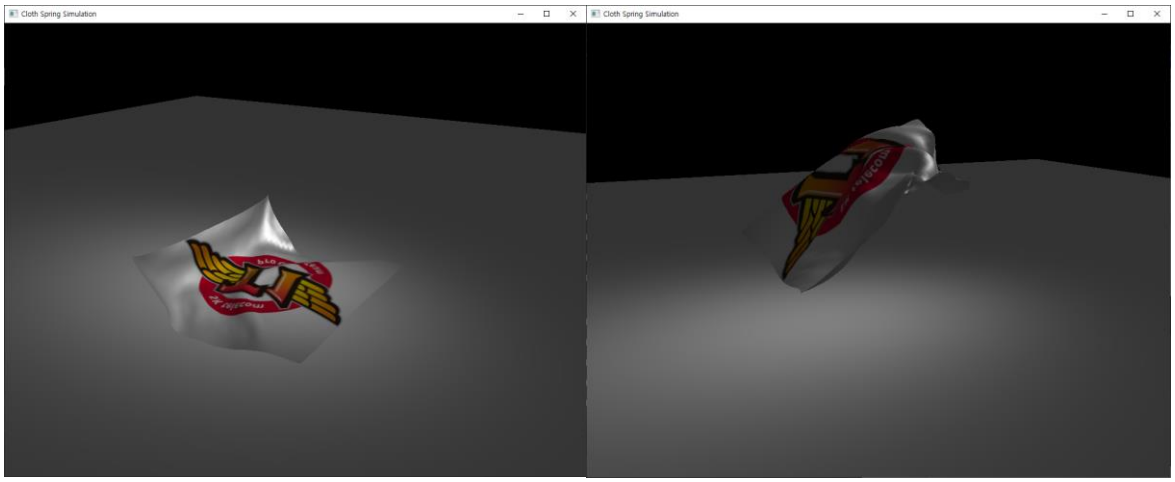
읽어온 texture를 그려줌

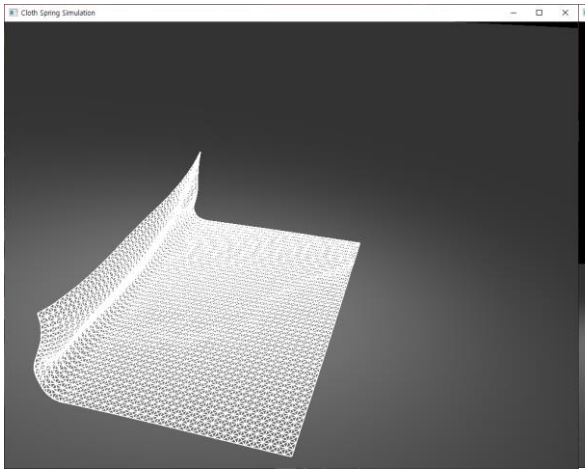
cloth.h – Gluint mass_cloth::LoadTexture(const char *filename)

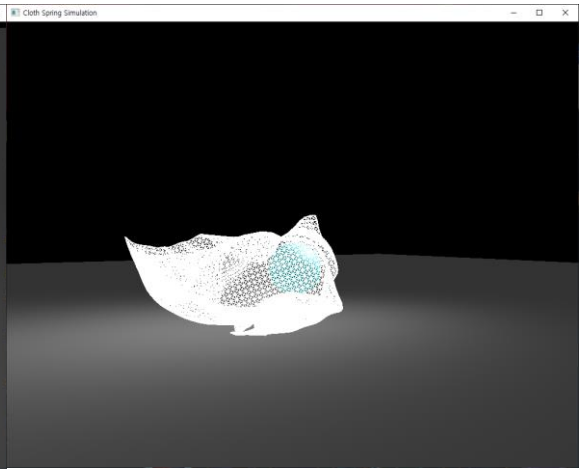
3. 핵심 소스코드

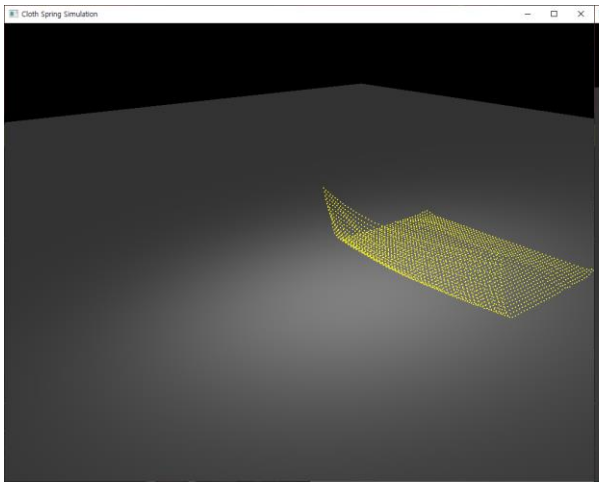
2. 구현 과정에 포함.

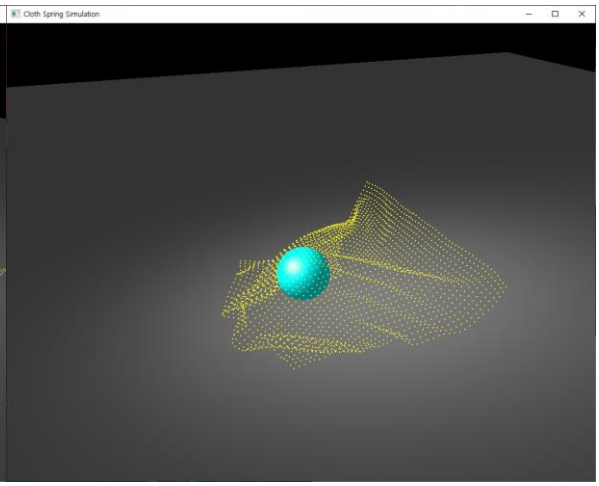
4. 구현 결과 화면




Cloth Spring Simulation


Cloth Spring Simulation


Cloth Spring Simulation


Cloth Spring Simulation