



# 3D 입체 모델링을 위한 OpenGL 기반 픽셀유동화

# CONTENTS

01 프로젝트  
개요

02 프로그램 소개

03 주요 기능  
소개

04 제한점

05 진행과정 및  
역할분배

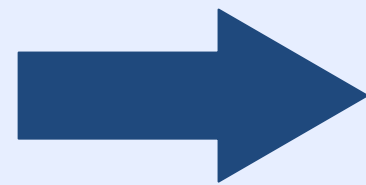
01

---

프로젝트

개요

# 개요

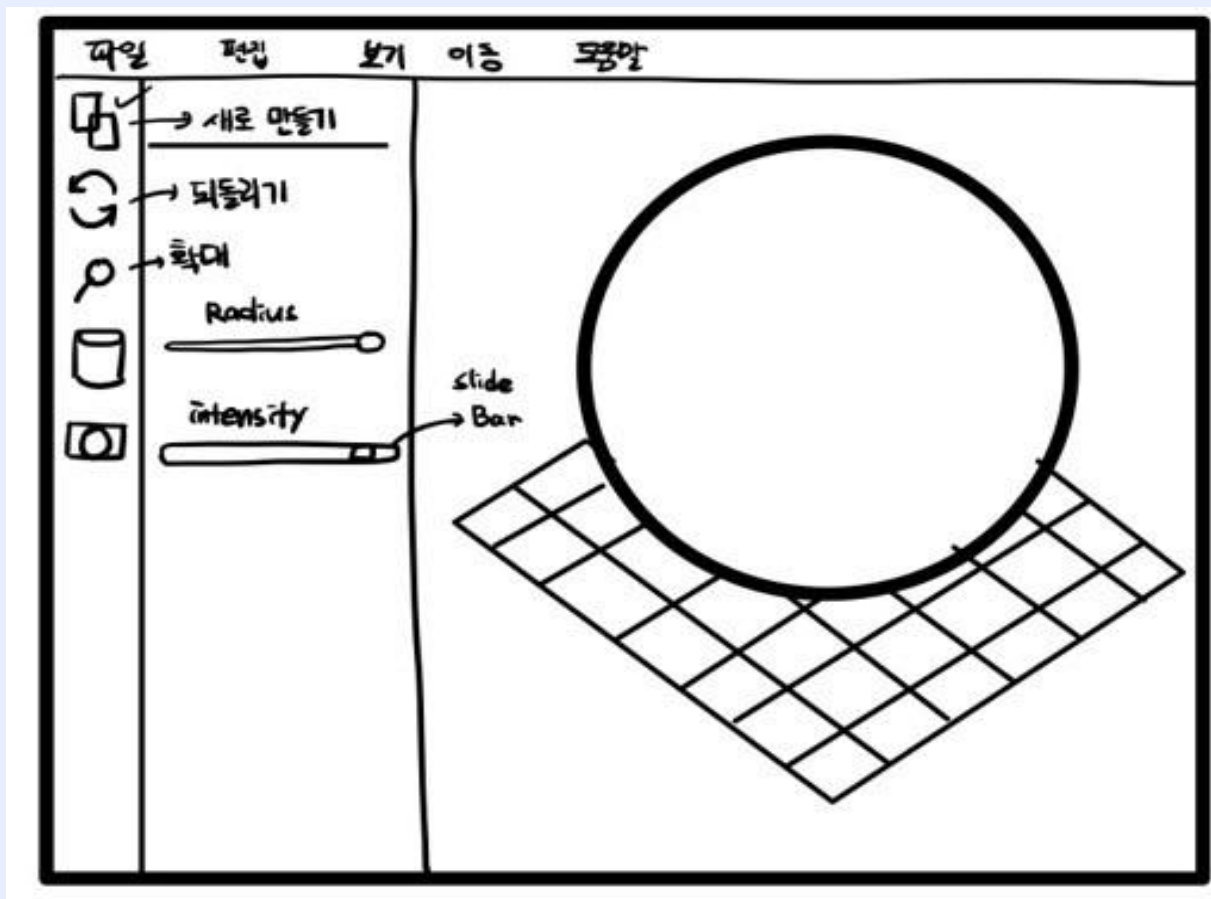


**픽셀유동화(Liquify) - 포토샵 등 기존 2D 그래픽물에 존재하는 도구**  
**이미지를 변형·보정하는 데 사용, 현재 2D에 한정되어 있음**

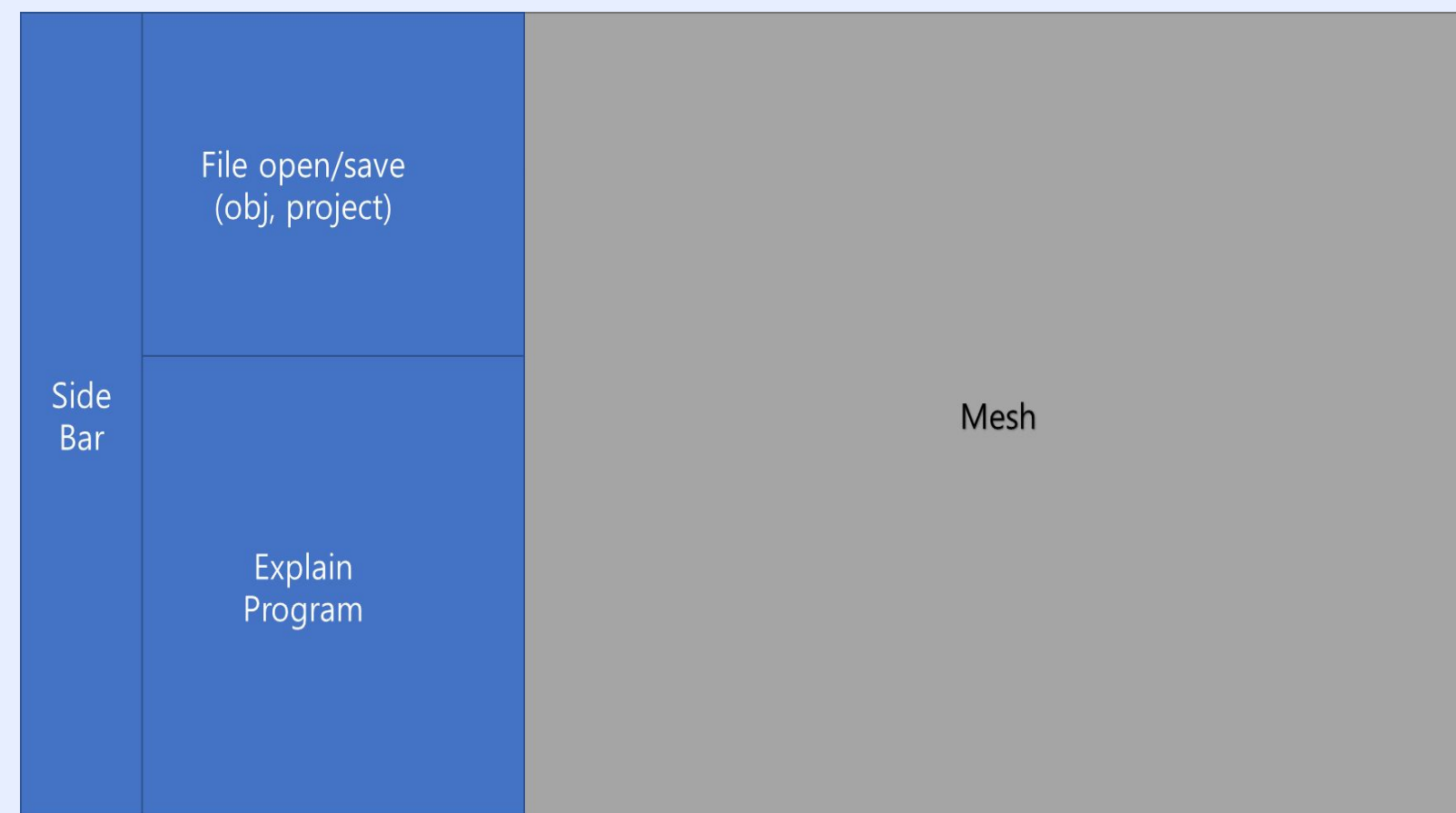
**이 기능을 3D 모델링에 적용 가능케 하는 것에 목적 : 3D 모델의 손쉬운 변형·보정을 위한**

# 개요

## 설계도



## 구성도



**목적 : 3D 스컬핑 프로그램에 픽셀유동화 기능을 하나의 툴로 포함**

02

---

**프로그램**

**소개**

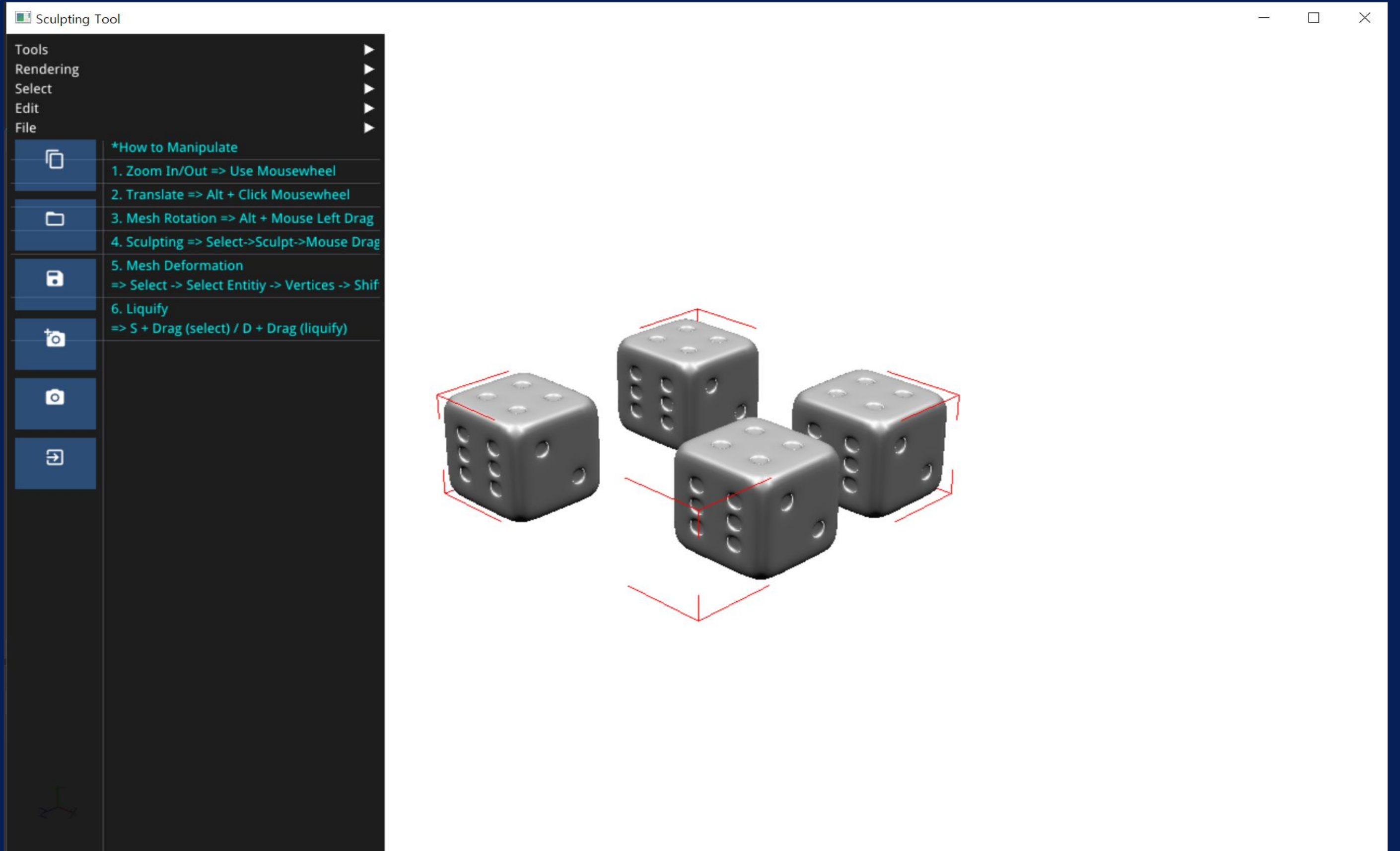
**02**

---

**1) GUI**



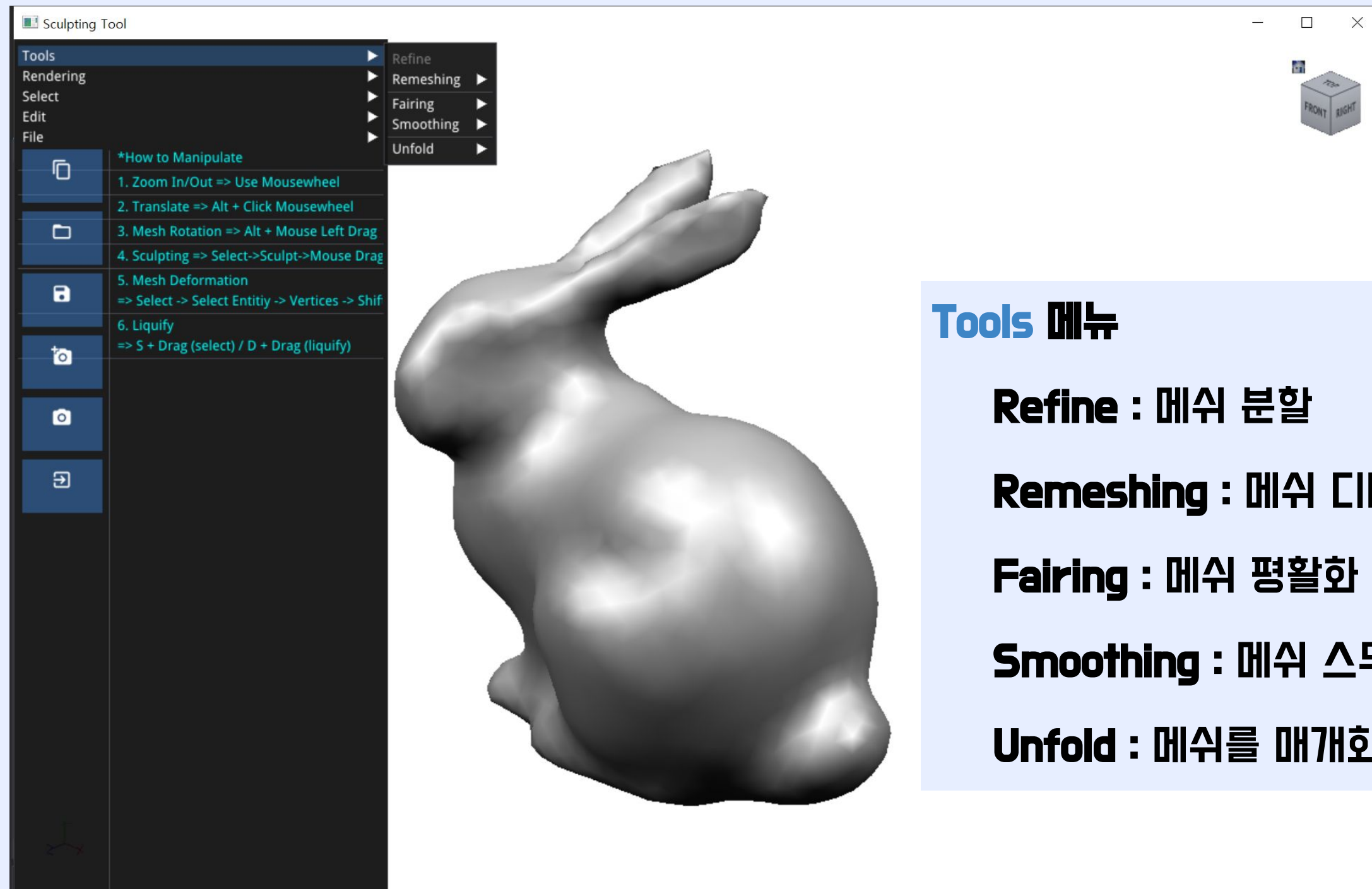
# 최종 UI





# GUI

## ImGui 메인메뉴



### Tools 메뉴

**Refine** : 메쉬 분할

**Remeshing** : 메쉬 디테일 강도 변화

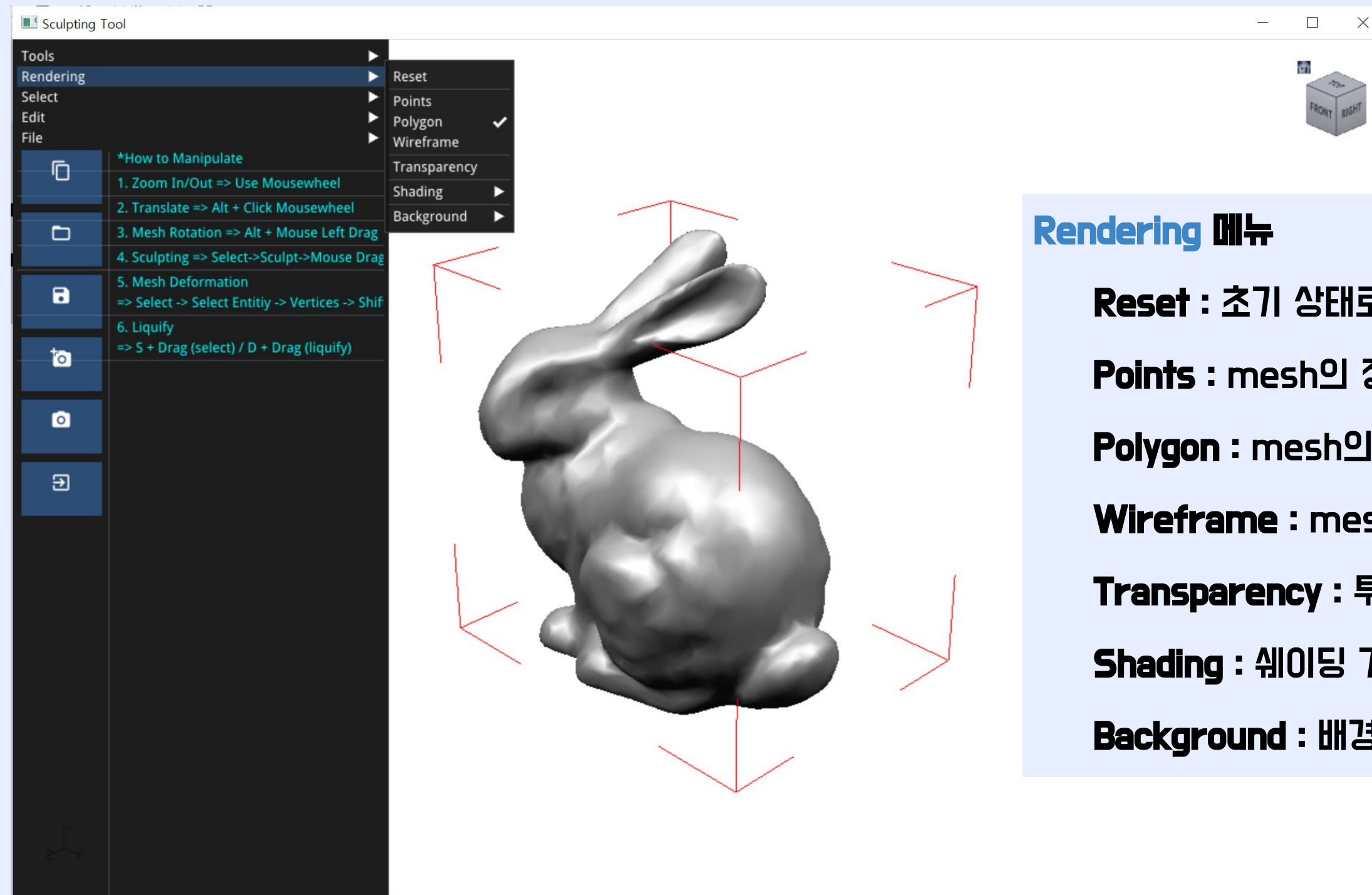
**Fairing** : 메쉬 평활화

**Smoothing** : 메쉬 스무딩

**Unfold** : 메쉬를 매개화 하는 메뉴

# GUI

## ImGui 메인메뉴



### Rendering 메뉴

**Reset** : 초기 상태로 되돌리기

**Points** : mesh의 정점 렌더링

**Polygon** : mesh의 폴리곤 렌더링

**Wireframe** : mesh의 edge 렌더링

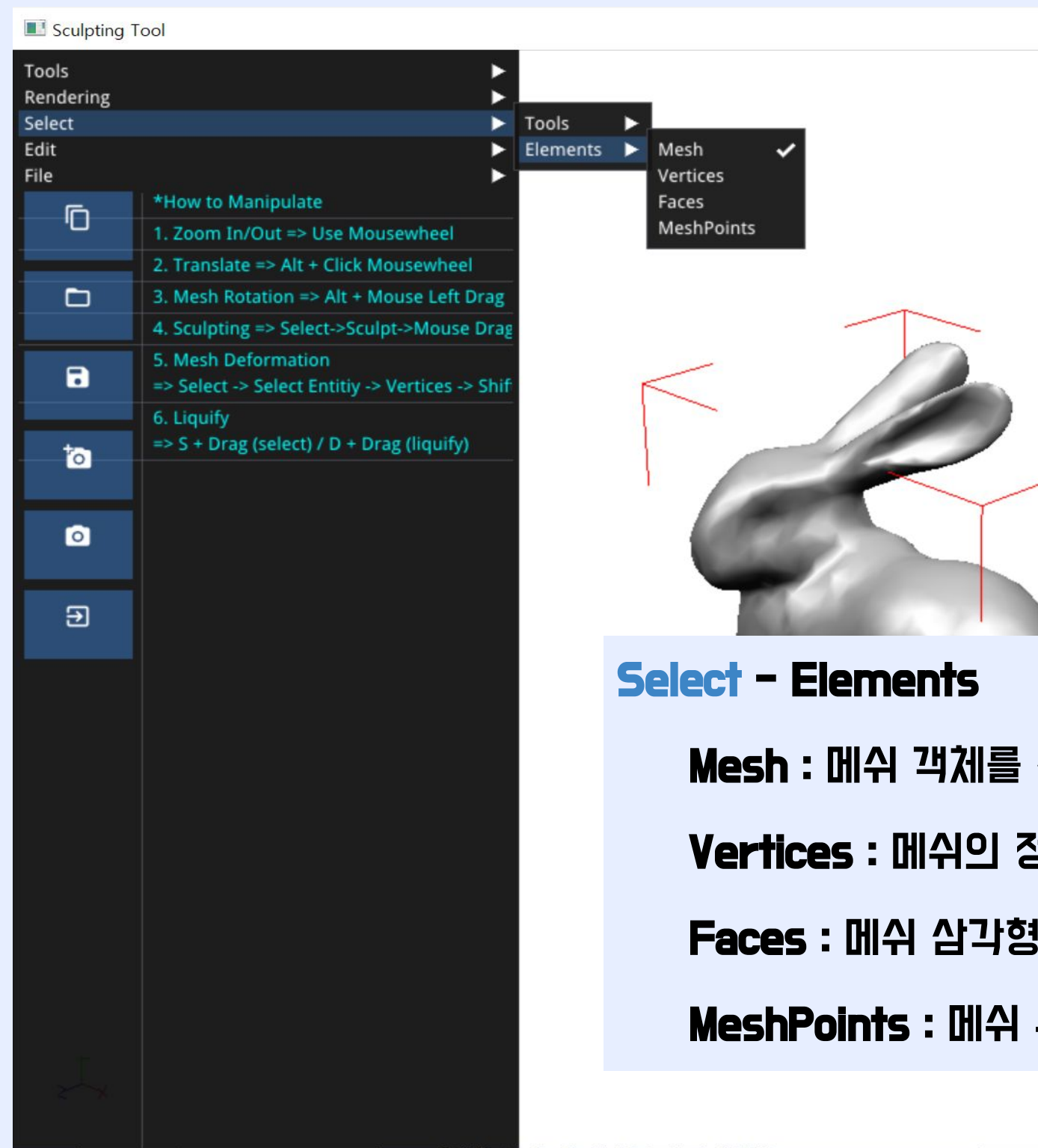
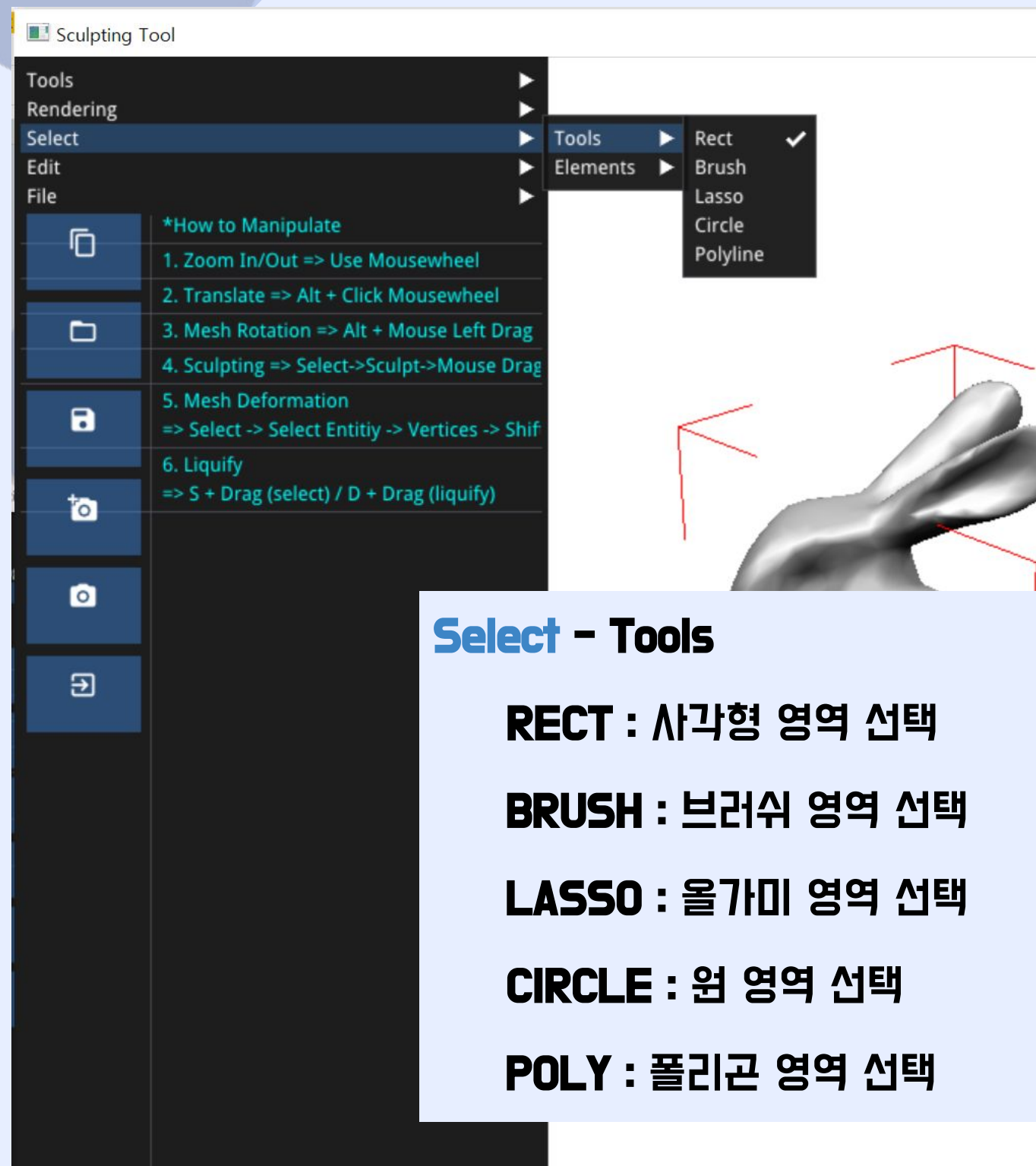
**Transparency** : 투과 여부 설정

**Shading** : 셰이딩 기법 변경

**Background** : 배경 Grid On/Off

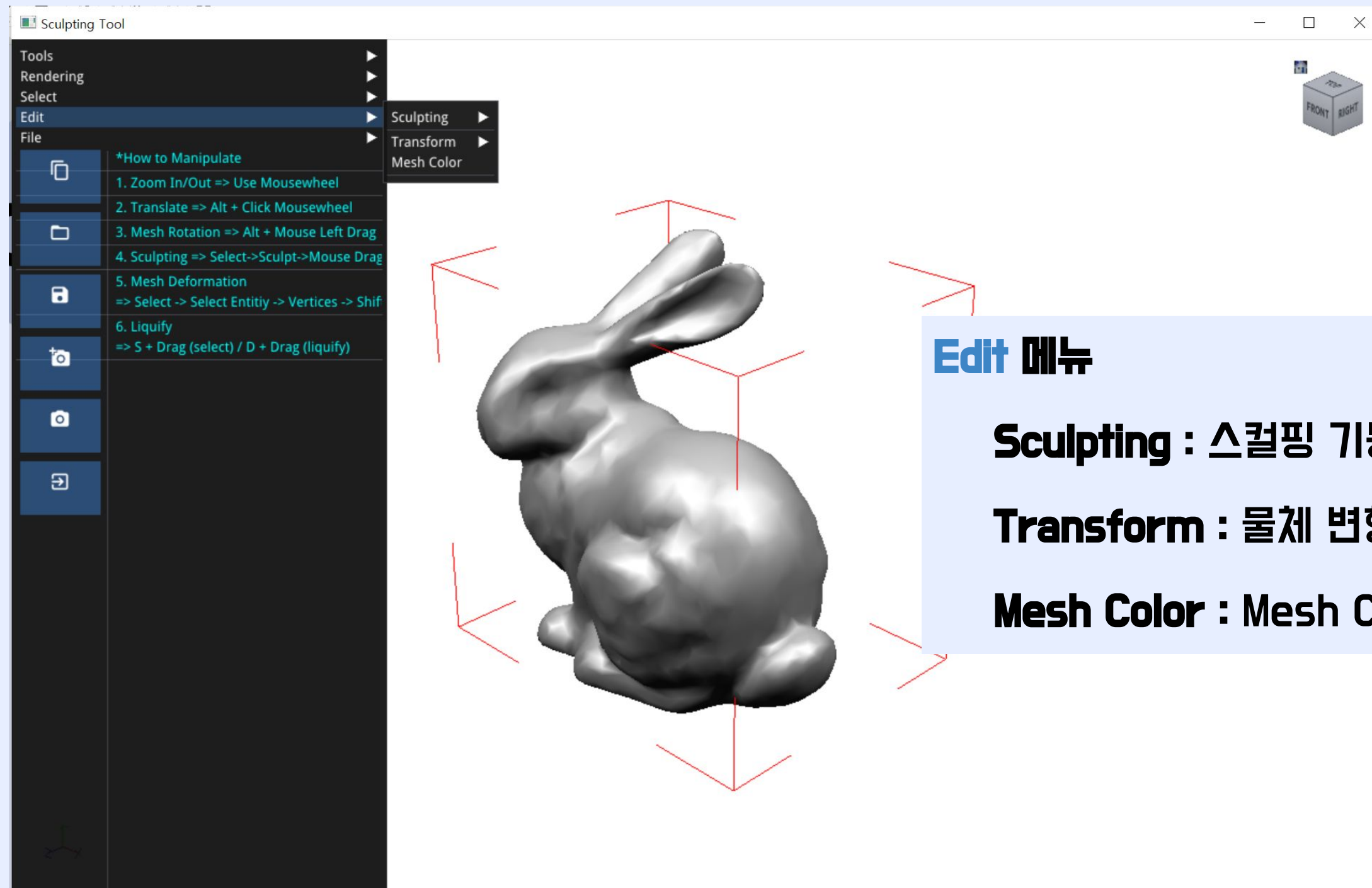
# GUI

## IMGUI 메인메뉴



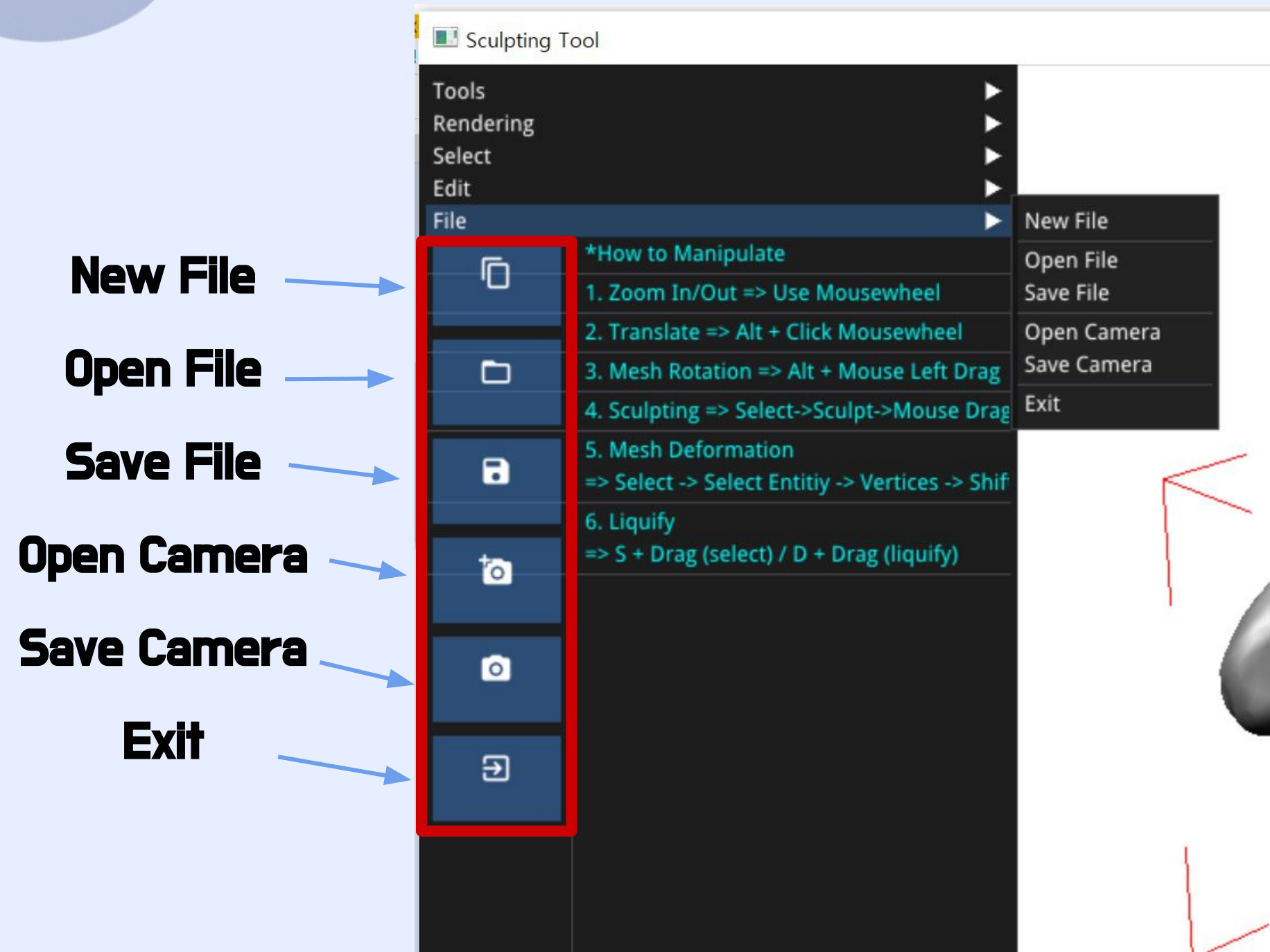
# GUI

## IMGUI 메인메뉴



# GUI

## ImGui 사이드 바 및 File 메뉴



**New File** : 현재 편집 중인 파일을 닫고 새로운 파일 오픈

**Open File** : .obj 파일 불러오기

**Save File** : .obj 파일 저장

**Open Camera** : .cam 파일(카메라 정보) 불러오기

**Save Camera** : 카메라 위치와 파일 저장

**Exit** : 프로그램 종료



**02**

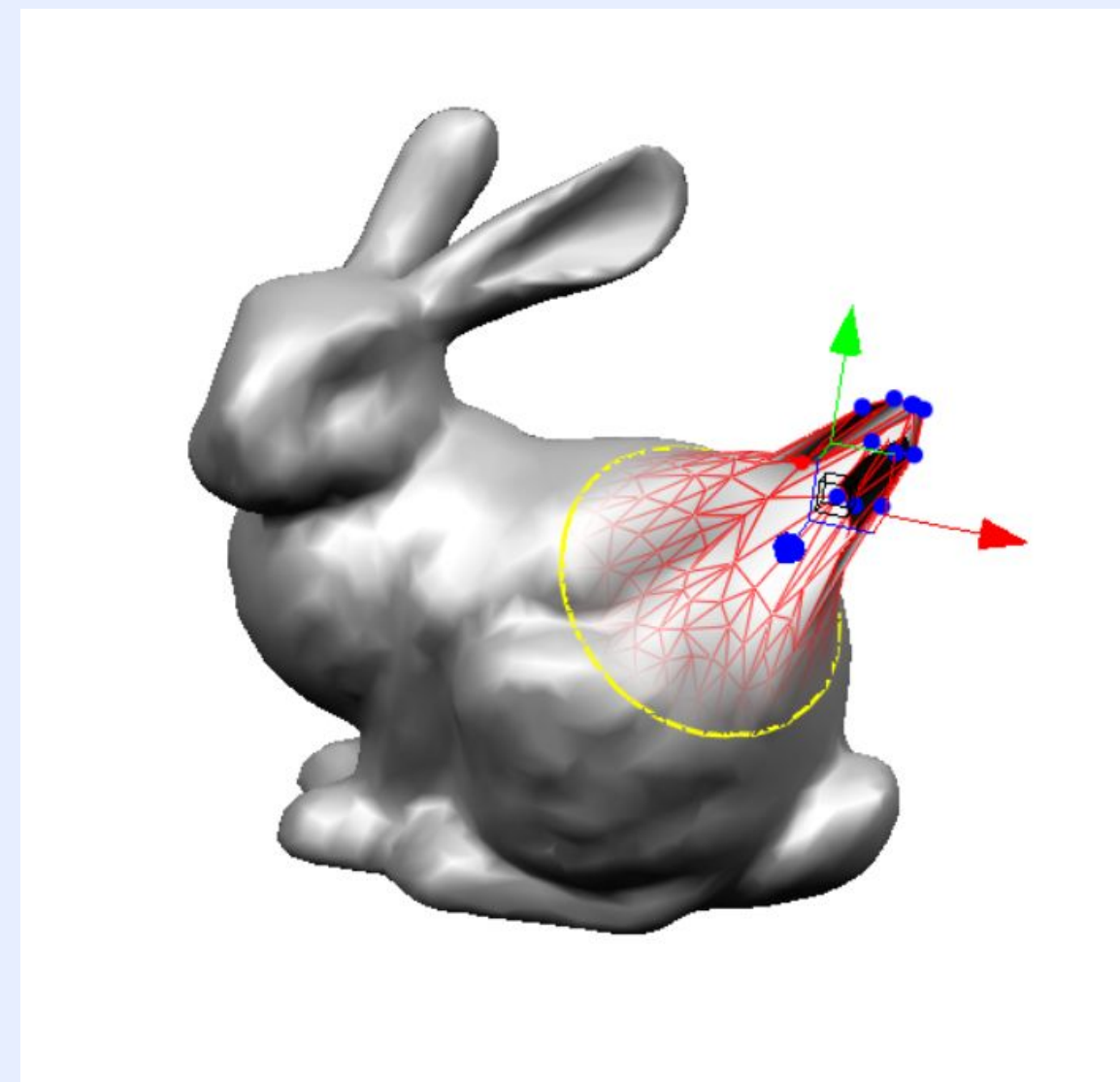
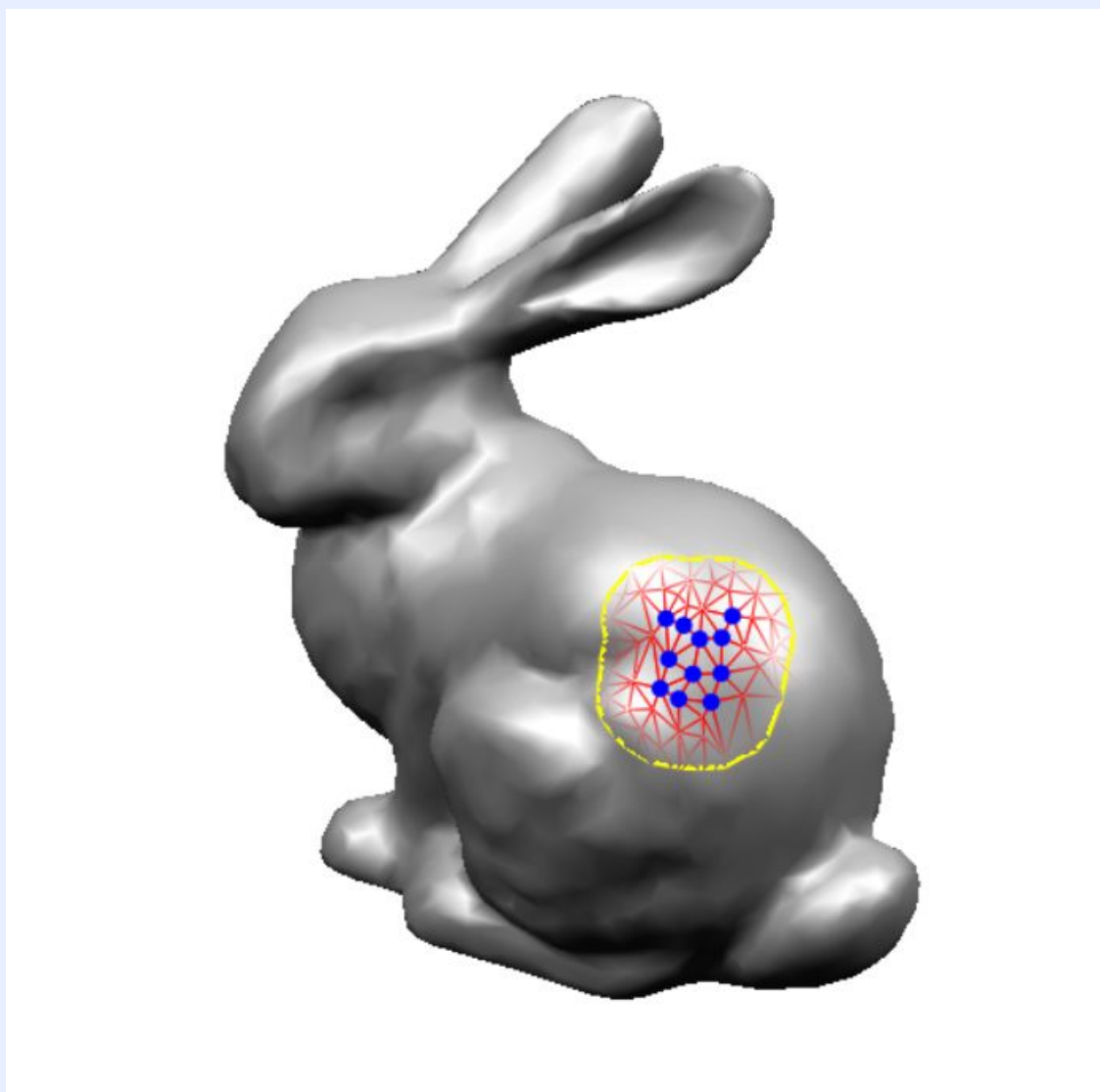
---

**2) Module**



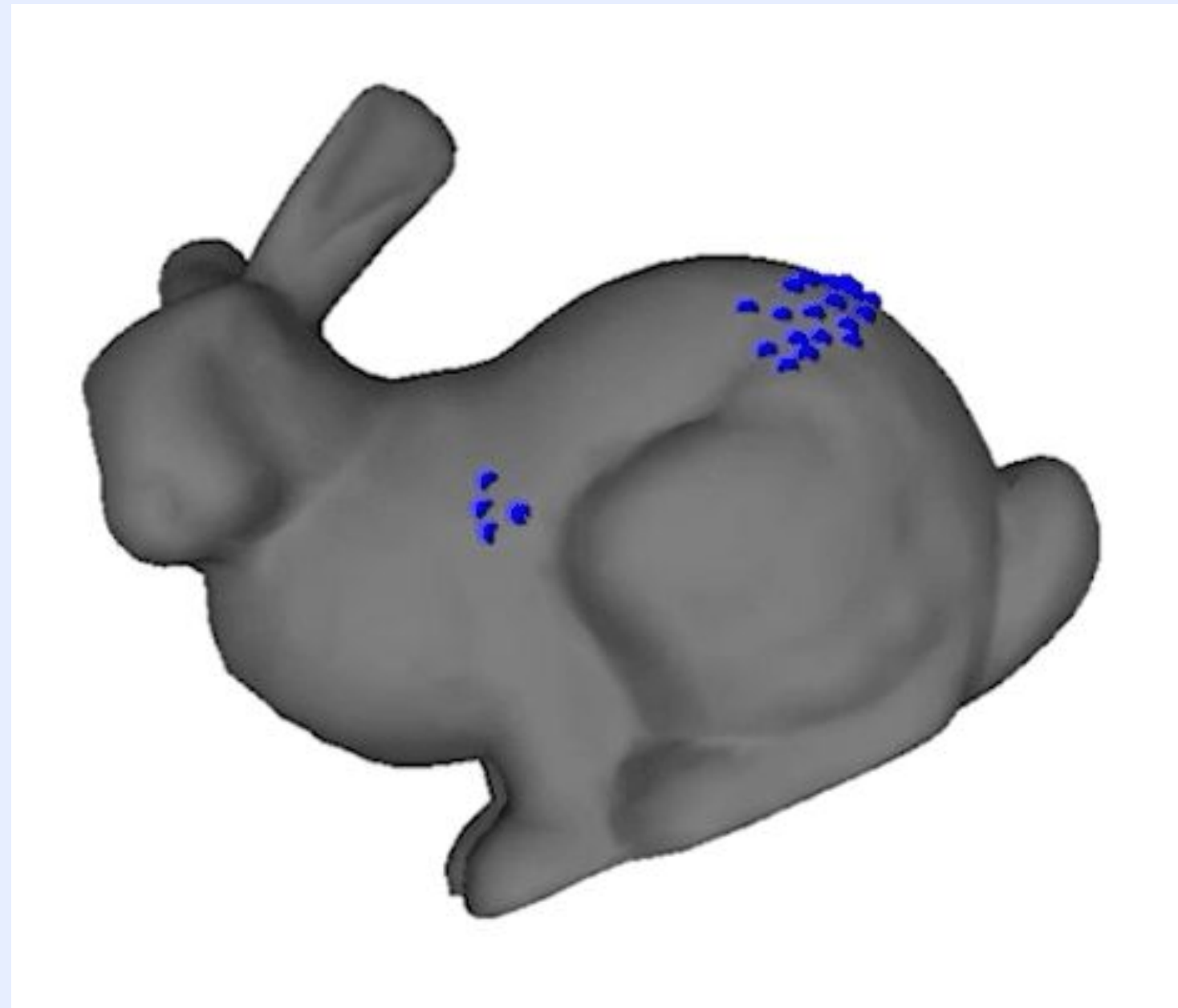
# 자유변형

## - 가중치를 이용한 자유 변형



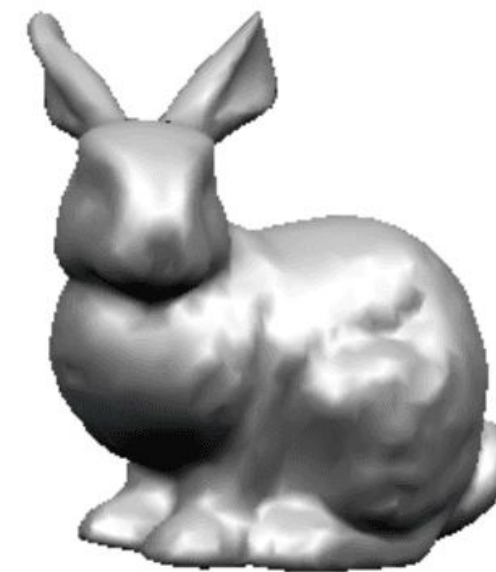
# 픽셀유동화

- mesh의 원형을 보존하는 상태에서 특정 방식으로 변형/보정



# Etc

- Zoom In/Out, Rotation 등의 기본 편의기능 포함



03

---

주요기능  
소개

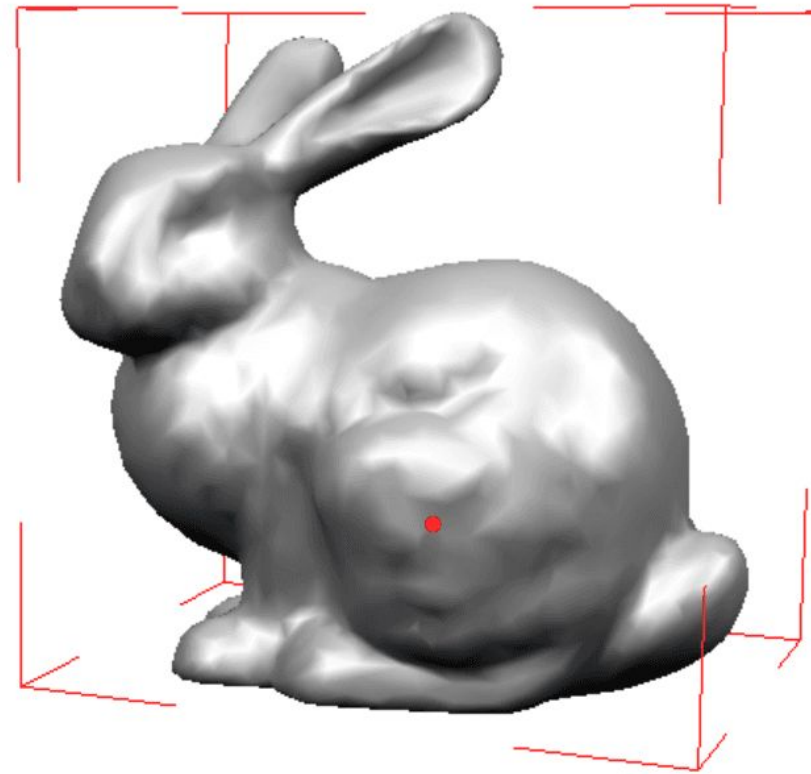
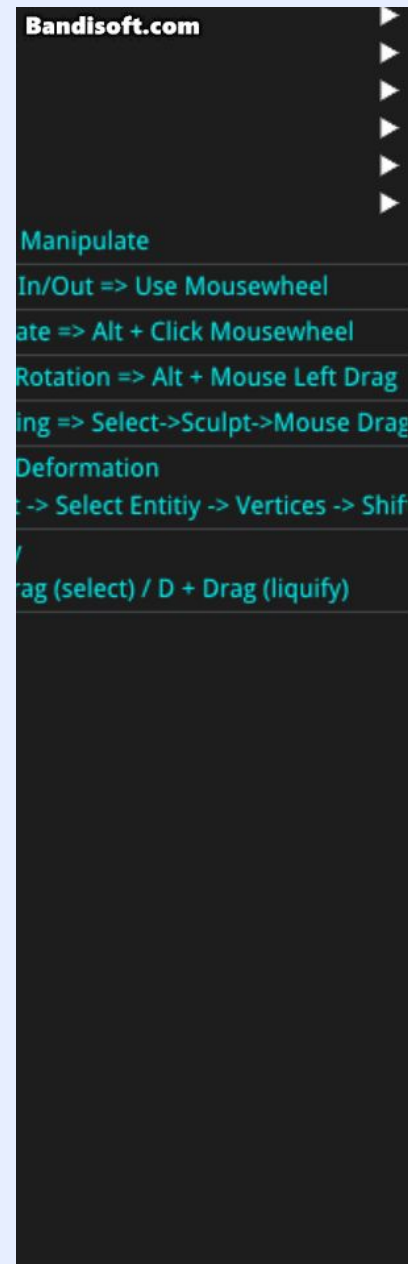
03

---

1) 기본기능

# OpenGL

## Drag - Mesh 원하는 부위의 Vertices 선택

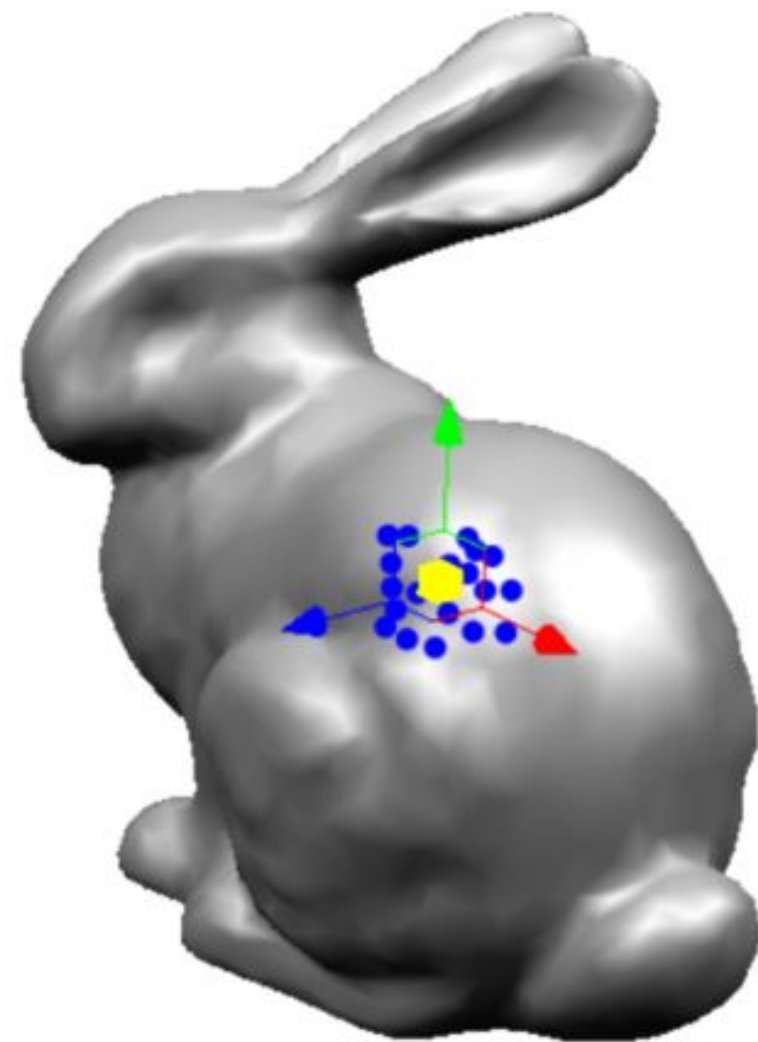




# OpenGL

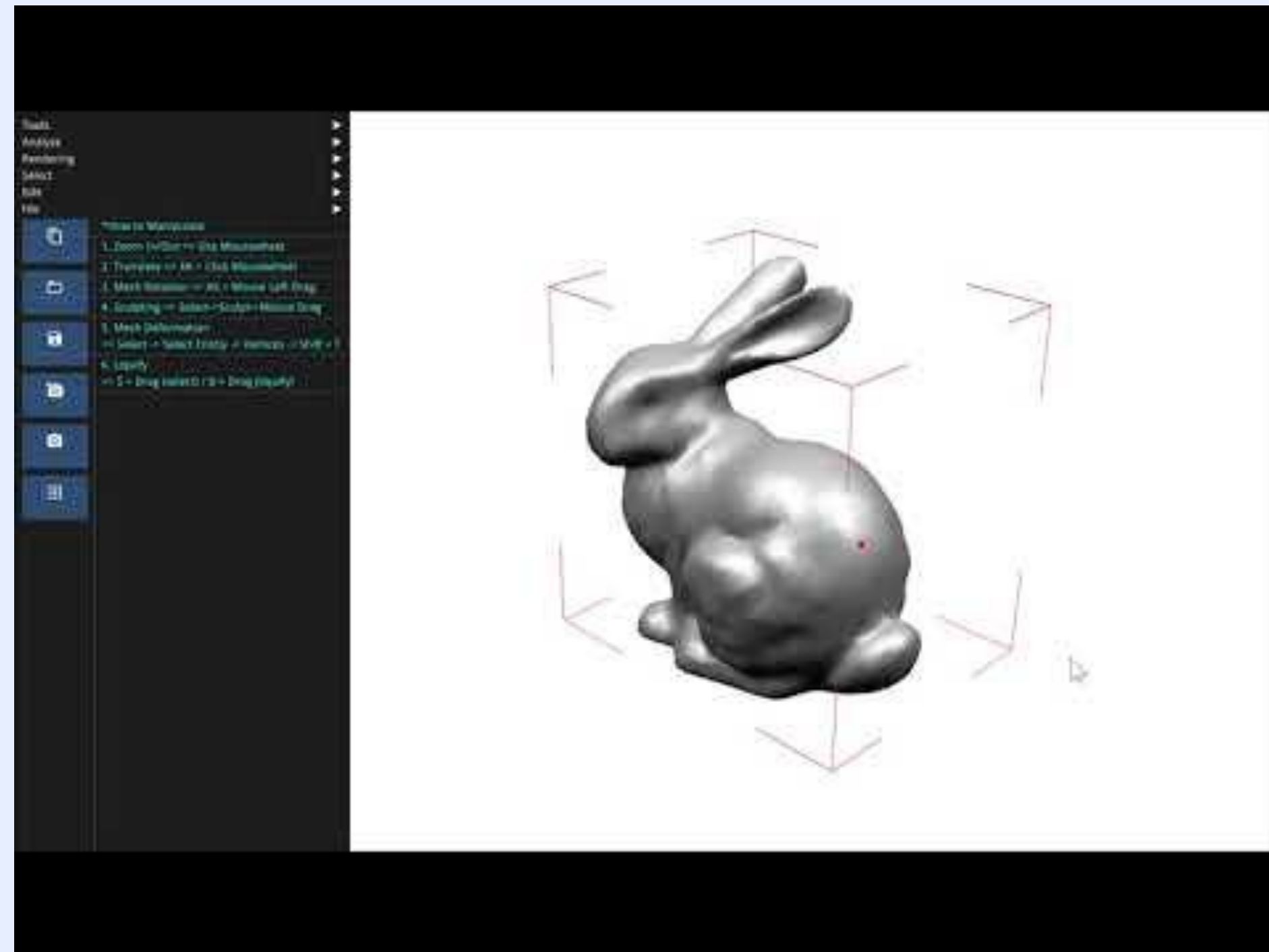
## 자유변형 - Shift + T

How to Manipulate  
Zoom In/Out => Use Mousewheel  
Translate => Alt + Click Mousewheel  
Mesh Rotation => Alt + Mouse Left Drag  
Sculpting => Select->Sculpt->Mouse Drag  
Mesh Deformation  
Select -> Select Entity -> Vertices -> Shift  
Liquify  
S + Drag (select) / D + Drag (liquify)



# OpenGL

자유변형 - Shift + T

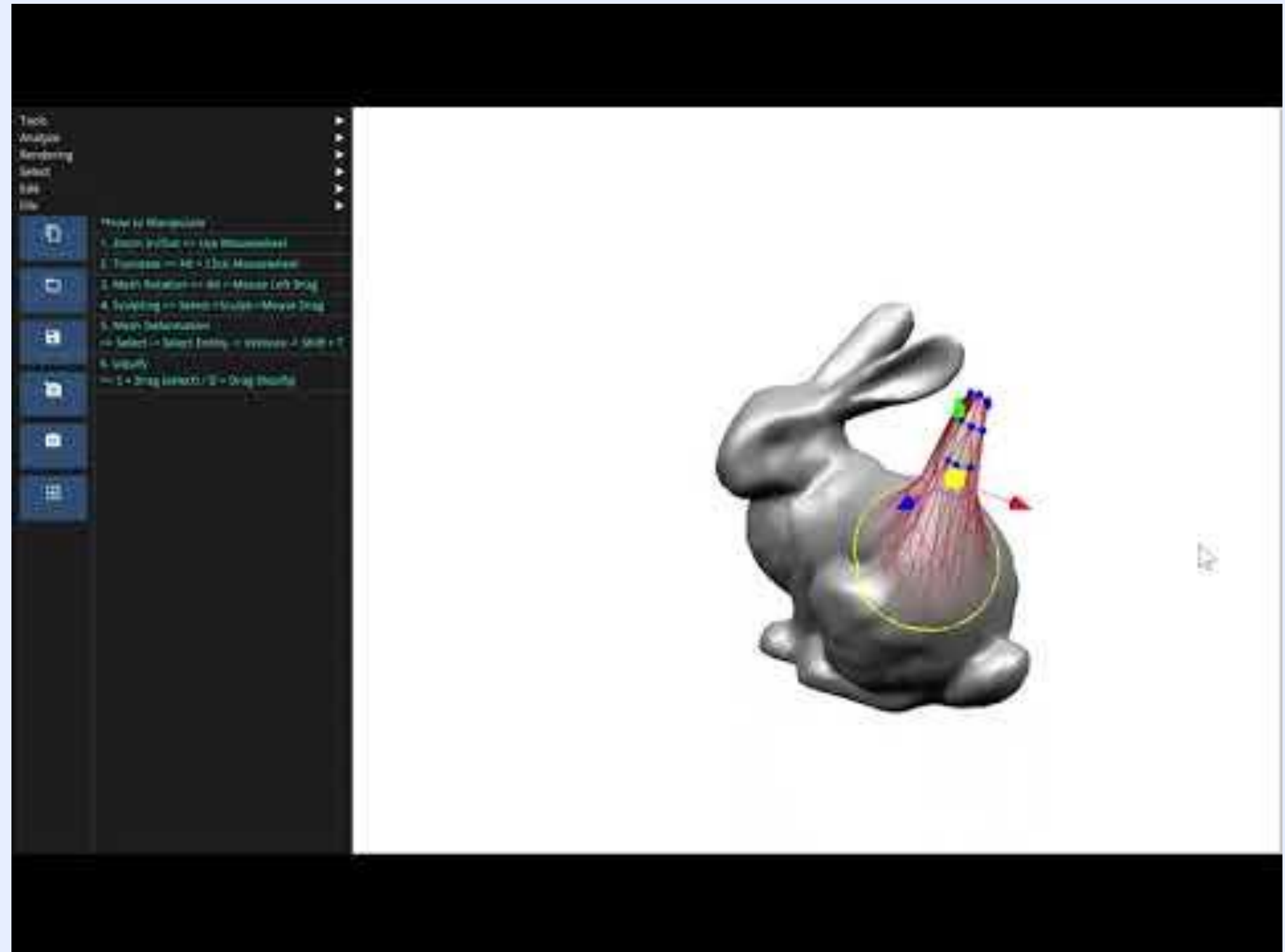


# OpenGL

## 자유변형 (2) - 변형 영역 확장 후 실행

### 가중치를 이용한 자유 변형


- ▶ Ctrl+Scroll을 통해 변형 영역 조절
- ▶ 정점의 중심에서 멀어질수록 변형 가중치 감소



**03**

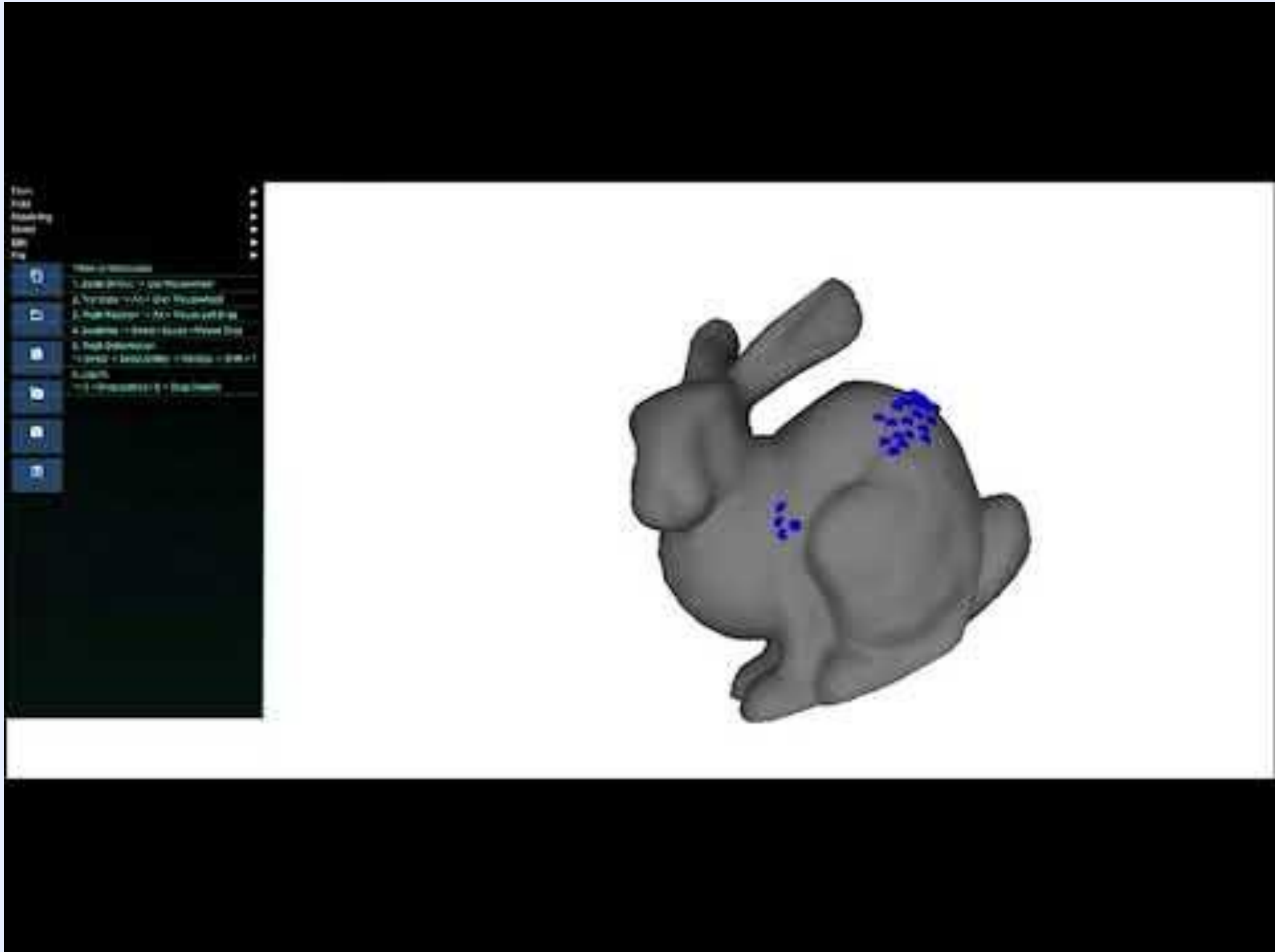
---

**2) Liquify**

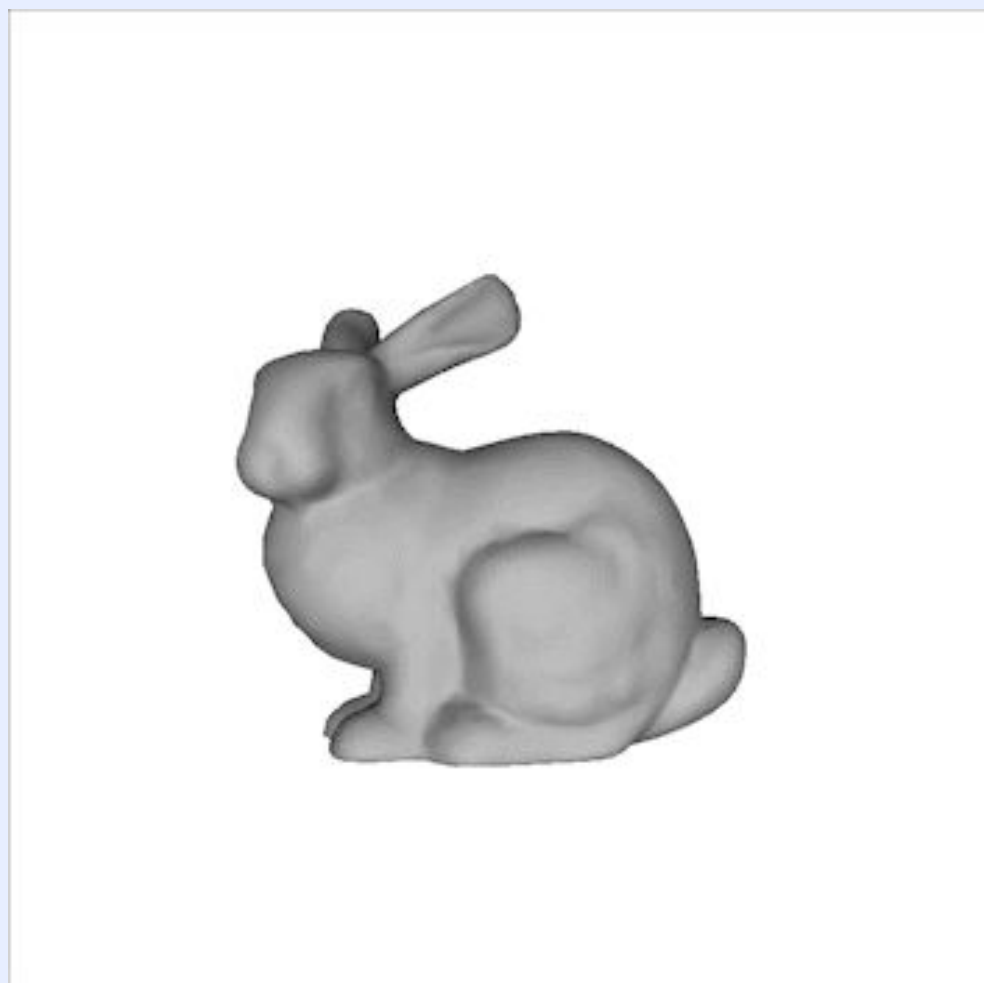
The image features the OpenGL logo, which consists of the word "OpenGL" in a bold, black, sans-serif font. The text is centered horizontally and is superimposed on a background of overlapping, semi-transparent geometric shapes in shades of light blue and grey. The shapes are arranged in a way that creates a sense of depth and movement, with some shapes appearing to be in front of others. The overall aesthetic is clean and modern, typical of a technical or scientific logo.

# OpenGL

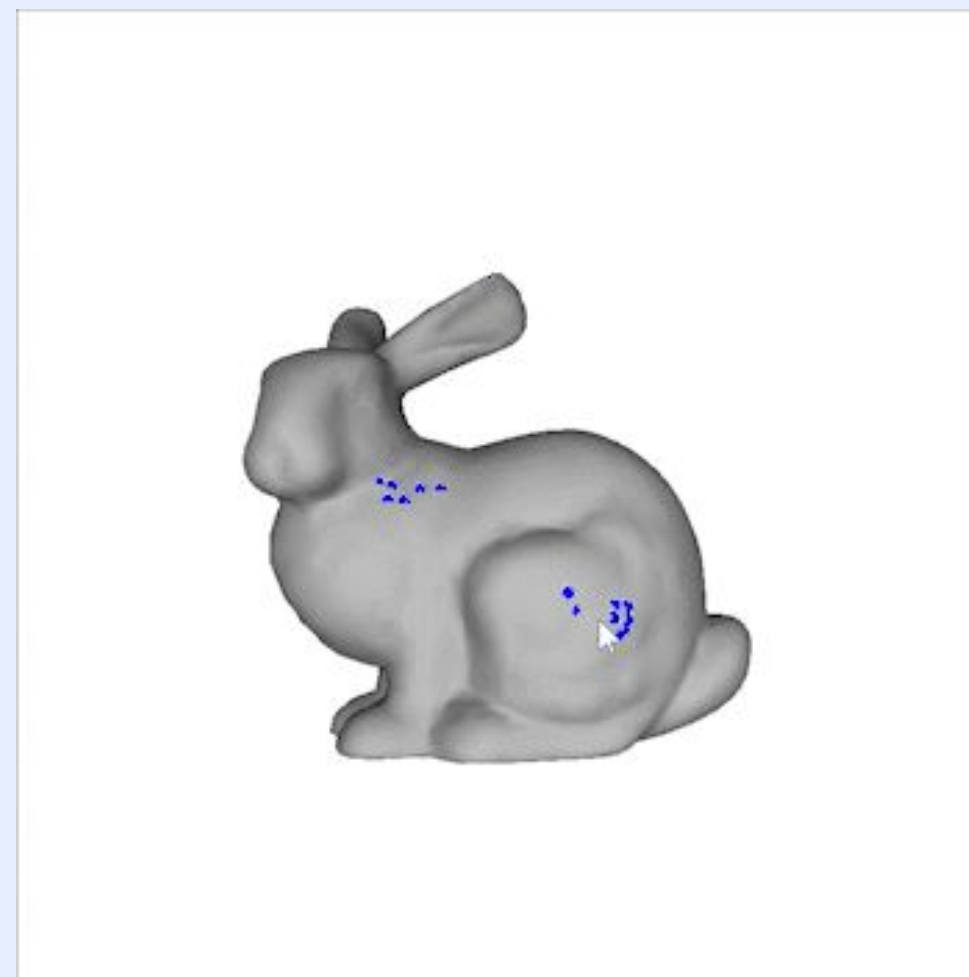
## 시연 영상



S + (Right Button) Drag => **Select mode**



D + (Right Button) => **Deformation mode**





### ARAP 기반 Polygon Mesh Deformation

- **Arap (As-Rigid-As-Possible Dynamic Deformation)**  
모델의 원형을 최대한 보존하면서 변형시키기 위해 폴리곤 메시 모델을 사면체 메시 모델로 변환하고, 사면체를 기본 Mesh 변형과 최대한 유사하게 변형 시키는 방법

$$E = \sum_{i=1}^n V_i \| M_i - \hat{M}_i \|_F^2$$

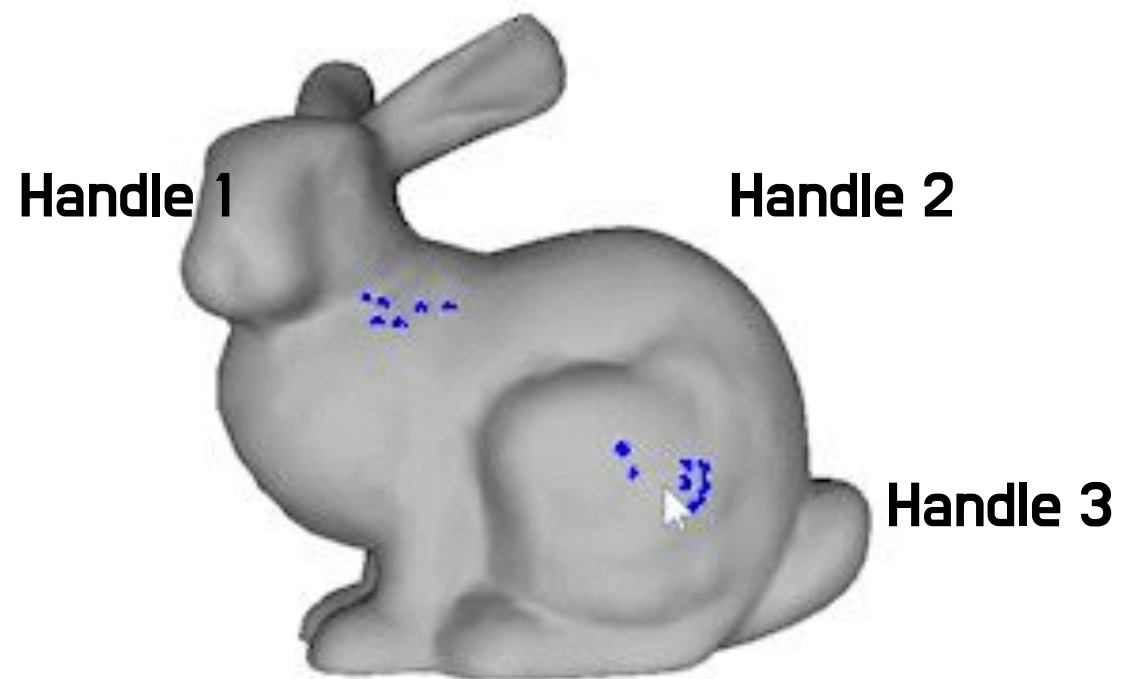
변형된 사면체들은 위 에너지 함수 식의 최소값을 만족

$n$  은 전체 사면체 수,  
 $V_i$  는 사면체 부피,  
 $M_i$  는 사면체의 이상변환행렬  
 $\hat{M}_i$  는 사면체의 실제변환행렬  
 $F$  는 Frobenius norm  
 $E$  는 변형된 사면체의  
 꼭짓점 좌표에 관한 이차 함수

### ARAP

변형 에너지를 최소화하는 과정에서  
제약조건으로 제어 정점을 설정





### ARAP

사용자가 직접 조작하는 Handle 이외의 Handle은 제어점점으로서 고정되어 작용 : 변형 에너지를 최소화하기 위함

#### Handle 1

- ▶ Handle 2, 3의 vertices는 고정된 상태로 Handle 1의 vertices만 이동됨

#### Handle 2

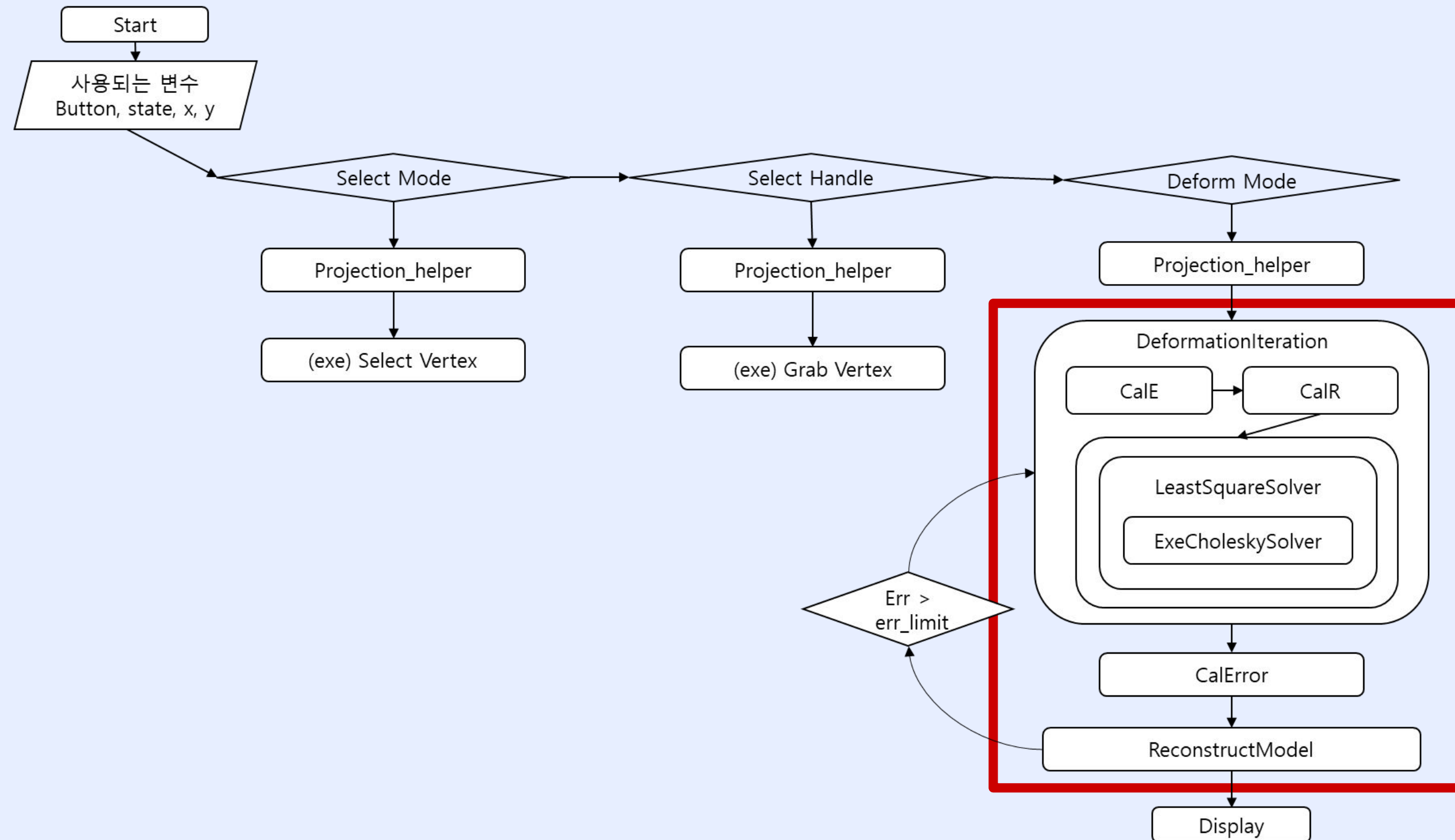
- ▶ Handle 1, 3의 vertices는 고정된 상태로 Handle 2의 vertices만 이동됨

#### Handle 3

- ▶ Handle 1, 2의 vertices는 고정된 상태로 Handle 3의 vertices만 이동됨

## 알고리즘 (Mouse Callback Function)

(각 함수를 통해 알고리즘이 동작되는 과정과 Callback 함수를 통해 실제 Mesh 변형에 반영되는 과정 보여 줌)



## 알고리즘 - ARAP 기반 형상변형

**요약 - LeastSquareSolver 반복 실행 => Error 갱신**

**=> error가 감소하며 변형 mesh의 좌표가 점진적으로 최적화됨**

(Mesh 변형이 Iterative하게 일어나기 때문에 변형 직후의 Mesh가 연속적으로 움직임을 시연 영상에서 확인할 수 있음)

```
Here is the matrix m:  
-0.997497  0.617481 -0.299417  
 0.127171  0.170019  0.791925  
-0.613392 -0.0402539  0.64568
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:1, err:0.177894
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:2, err:0.083912
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:3, err:0.049332
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:4, err:0.035094
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:5, err:0.029172
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:6, err:0.021457
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:7, err:0.021040
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:8, err:0.015068
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:9, err:0.012159
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:10, err:0.012159
```

```
-----  
Start to solve LeastSquare method by Cholesky solver  
-----  
iteration:11, err:0.008032
```

### Mouse Callback Function

- Deform Mode + 우클릭 드래그 입력 시 실행
- 변형 좌표 계산 ▶ err 계산 ▶ mesh 재구축
- err가 충분히 작아질 때까지 Iteration 반복

```
while (err > err_limit)
{
    iteration++;
    vector<double> res = DeformationIteration();
    CalError(res);
    mesh = ReconstructModel(mesh, res);
    printf("iteration:%d, err:%f\n\n", iteration, err);

    Display();
}
```



### Deformation Iteration

- LeastSquareSolver의 실행을 위해  
사용자가 지정한 제어 정점과  
직전 Iteration에서의 회전변환행렬(R) 정보,  
기존 Mesh의 정보를 계산, 전달
- 정점의 위치를 고정시킨 후  
회전변환행렬 계산

```
for (int i = 0; i < mesh->numvertices; ++i)
{
    b_temp = Eigen::Vector3f::Zero();
    for (auto iter = connectedMap[i + 1].begin(); iter != connectedMap[i + 1].end(); ++iter)
    {
        eij = Eigen::Vector3f(originMesh->vertices[3 * (i + 1) + 0] - originMesh->vertices[3 *
(*iter) + 0],
                             originMesh->vertices[3 * (i + 1) + 1] - originMesh->vertices[3 * (*iter) + 1],
                             originMesh->vertices[3 * (i + 1) + 2] - originMesh->vertices[3 * (*iter) + 2]);

        Ri = R[i];
        Rj = R[*iter - 1];
        b_temp += 0.5f * (Ri + Rj) * eij;
    }
    b_top.push_back(b_temp);
}
```

```
res = LeastSquareSolver(controlIndices, connectedMap, b_top, originMesh);
```

# OpenGL

## 알고리즘 - 함수 설명

### LeastSquareSolver

회전변환행렬 고정시킨 상태에서  
각 정점이 제약조건\*을 만족시키면서 에너지를 최소화하기 위해  
이동되어야 할 좌표를 계산함.

(\* 제약조건: 사용자가 select한 제어 정점의 좌표)

```
// Laplacian
int row = 0;
for (auto iterMap = ++connectedMap.begin(); iterMap != connectedMap.end(); ++iterMap, ++row)
{
    int col = row;

    tripletListA.push_back(Eigen::Triplet<double>(row, col, (*iterMap).second.size()));
    for (auto iterSet = (*iterMap).second.begin(); iterSet != (*iterMap).second.end(); ++iterSet)
    {
        col = (*iterSet) - 1;

        value = Eigen::Triplet<double>(row, col, -1.0f);
        tripletListA.push_back(value);
    }
}
```

```
// Constraint
row = 0;
for (auto iter = controlIndices.begin(); iter != controlIndices.end(); ++iter, ++row)
{
    value = Eigen::Triplet<double>(mesh->numvertices + row, *iter - 1, 1);
    tripletListA.push_back(value);

    for (int i = 0; i < 3; ++i)
    {
        value = Eigen::Triplet<double>(mesh->numvertices + row, i, mesh->vertices[*iter * 3 + i]);
        tripletListb.push_back(value);
    }
}
```

# OpenGL

## 알고리즘 - 함수 설명

### ExeCholeskySolver

(콜레스키 분해)

(과정)

1. 최소행렬 계산

행렬 계산에 Eigen SparseMatrix 라이브러리 사용

2. 콜레스키 분해 진행

$$A = LL^*$$

```
Eigen::MatrixXd ExeCholeskySolver(Eigen::SparseMatrix<double>* A, Eigen::SparseMatrix<double>* b)
{
    Eigen::SimplicialCholesky<Eigen::SparseMatrix<double>> solver(*A);

    cout << "-----" << endl;
    cout << "Start to solve LeastSquare method by Cholesky solver" << endl;
    cout << "-----" << endl;

    Eigen::MatrixXd res = solver.solve(*b);

    return res;
}
```

=> error 감소시키기 위한 최소자승해 계산을 위해 해당 식 계산 (전처리 단계)



# OpenGL

## 알고리즘 - 함수 설명

### CalE

- 정점 값을 계산

```
vector<vector<vector3>> CalE(_GLMmodel* iMesh)
{
    vector<vector<vector3>> res;

    for (int i = 0; i < iMesh->numvertices; ++i)
    {
        res.push_back(vector<vector3>());
        for (auto iter = connectedMap[i].begin(); iter != connectedMap[i].end(); ++iter)
        {
            vector3 v = (vector3(
                iMesh->vertices[3 * (i + 1) + 0] - iMesh->vertices[3 * (*iter) + 0],
                iMesh->vertices[3 * (i + 1) + 1] - iMesh->vertices[3 * (*iter) + 1],
                iMesh->vertices[3 * (i + 1) + 2] - iMesh->vertices[3 * (*iter) + 2]));
            res[i].push_back(v);
        }
    }

    return res;
}
```

### CalR

- (자코비안 행렬로) 비선형적인 변화 => 선형적 변화로 계산  
(이를 통해 deformationIteration 함수에서 최소자승해 계산을 가능하게 함)

```
Eigen::Matrix3f CalR(int index)
{
    Eigen::Matrix3f R;
    Eigen::MatrixXf eij(1, 3), eij_p(1, 3);
    Eigen::Matrix3f Si = Eigen::Matrix3f::Zero();

    for (int j = 0; j < connectedMap[index].size(); ++j)
    {
        for (int x = 0; x < 3; ++x)
        {
            for (int y = 0; y < 3; ++y)
            {
                Si(x, y) += e[index][j][x] * e_p[index][j][y];
            }
        }
    }

    Eigen::JacobiSVD<Eigen::MatrixXf> svd(Si, Eigen::ComputeFullU | Eigen::ComputeFullV);
    Eigen::MatrixXf U = svd.matrixU();
    Eigen::MatrixXf V = svd.matrixV();
    Eigen::MatrixXf S = svd.singularValues();

    R = V * U.transpose();

    return R;
}
```

# OpenGL

## 알고리즘 - 함수 설명

### CalError

(에러 계산)

에너지 최소화를 목적으로 함.

=> 최솟값을 갱신하는 iteration을 수행

=> 이 과정을 통해 점진적으로 Mesh 변형이 진행됨

```
void CalError(vector<double> solveVertices)
{
    float err_tmp = 0;

    for (int i = 0; i < solveVertices.size(); ++i)
    {
        err_tmp = max(fabs(mesh->vertices[i + 3] - solveVertices[i]), err_tmp);
    }

    err = min(err_tmp, err);
}
```

04

---

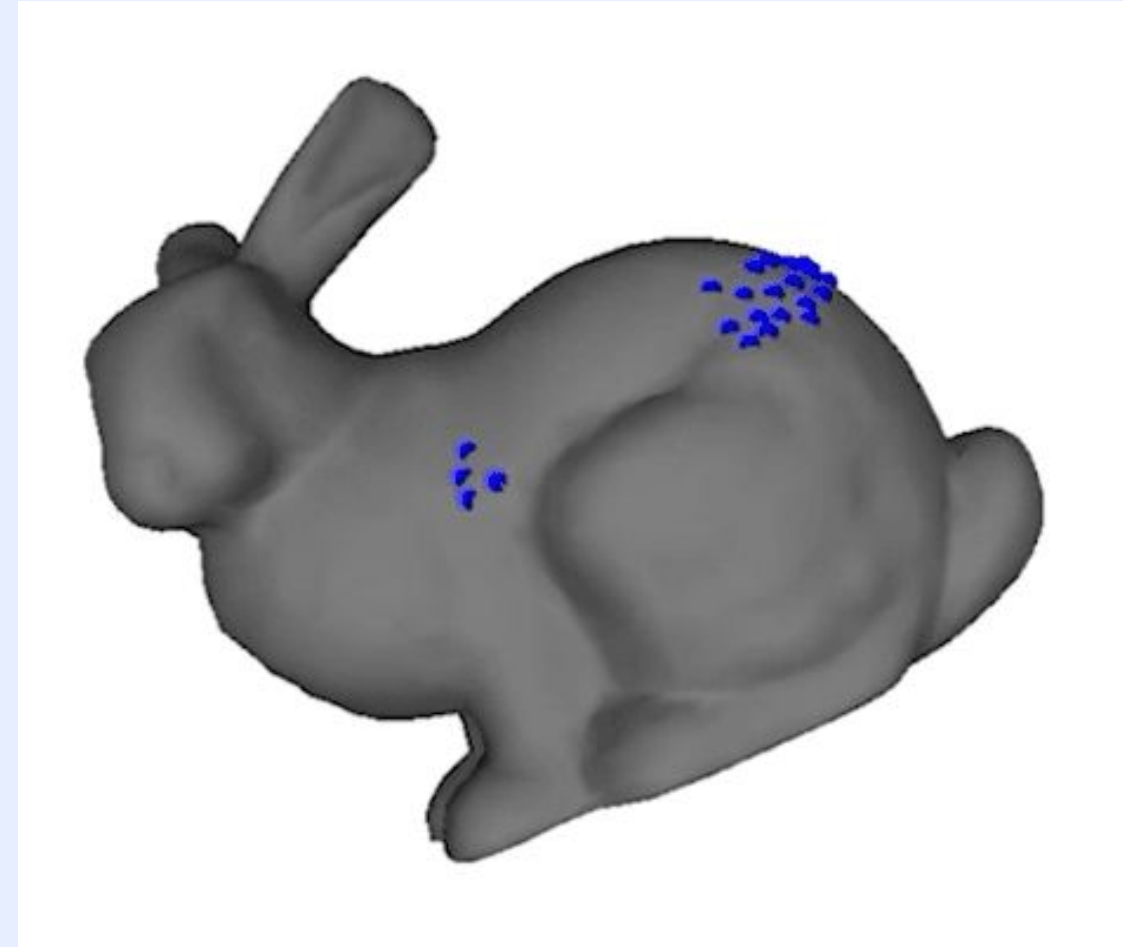
**제한점**

# 제한점

초기 의도



개발 결과



초기 의도는 부분적인 변형이었으나 개발 결과 전체적인 범위로 변형이 적용됨

▶ 차후 알고리즘 적용 방식을 확장하면 부분 변형 또한 실행가능할 것으로 기대



05

---

**1) 진행과정**

# 진행과정



# 진행과정

2-4주차 : 아이디어 도출 및 기획



2-3주차

## 0308 - 0321 (2~3주차)

- 기획안 제시
- 자유주제 아이디어 도출 및 정리
- 기획안, 구성도, 설계도 작성

3, 4



4주차

## 0321 - 0328 (4주차)

- 픽셀유동화 확장 방식 논의
- (Module) 소스코드/알고리즘 참고해서 방향성 확립
- (GUI) GUI/openGL 실행파일에 생성
- (GUI) GUI 프로그램 회의.

# 진행과정

5-8주차 : Test Program 개발, 오류 픽스



5-6주차

**0330 - 0412 (5-6주차)**

**Test Program 개발 시작**

- (Module) obj파일 로딩, 기본 기능 구현 (Zoom, Translate, Rotation)
- (GUI) GUI/openGL 실행파일에 생성
- (GUI) 메뉴 바, 사이드 바, 메인 편집 바 생성

3,4



7-8주차

**0412 ~ 0425 (7-8주차)**

- (Module) Obj 파일 로드 및 렌더링 알고리즘 개선
- (Module) 조명모델 추가
- (GUI) 모듈과 GUI 툴 박스의 기본 기능 연동 (Zoom, Translate, Rotation)
- (GUI) Test Program UI 개선

# 진행과정

## 9-14주차 : 최종 Sculpting Tool 프로그램 개발 (Liquify 모듈)



9-11주차

**0426 ~ 0517 (9-11주차)**

### **Sculpting Tool 개발 시작**

- (Module) Mesh 변형 및 스컬핑 기능 구현 (계속)
- (Module) Obj 로드 알고리즘 / 조명 모델 개선
- (GUI) 폴리곤 선 ON/OFF 기능 구현
- <sup>3,4</sup> - (GUI) 프로젝트(obj 파일) 메뉴에서 불러오기 연동 (메뉴 바)



12-14주차

**0518 ~ 0607 (12-14주차)**

- (Module) 픽셀유동화 알고리즘 조사 및 구현
- (Module) Mesh 변형, Mesh 단순화 기능 구현
- (GUI) Mesh 변형, 픽셀유동화 모듈 GUI와 연동 (메뉴 바)
- (GUI) 새 파일, 파일 열기 및 저장, 카메라 열기 및 저장, 종료 연동 (사이드 바)

# 진행과정

15-16주차 : SculptingTool 개발 마무리, 최종 발표



15-16주차

**0608 ~ 0621 (15-16주차)**

- (Module+GUI) Module + GUI 연동 오류 픽스
- 픽셀유동화 + 기본 기능 프로그램 병합

05

---

2) 역합분배



# 역할 분배



**Module  
& Liquify**

박예원



**Module  
& Liquify**

안준혁



**GUI  
& Liquify**


조예진



**GUI  
& Liquify**

안선영

# 역할 분배



## Module &Liquify

박예원 (조장)

### 프로그램 기획

- 기획안 제시 (픽셀유동화, 알고리즘)
- 기능 세부사항 수립


### Module (Test Tool)

- Rotation(시점 회전), Zoom in/out
- obj 생성&로딩 구현 및 개선
- 조명 모델 구현

### Sculpting, Liquify (Sculpting Tool)

- (알고리즘) 픽셀유동화 알고리즘 연구 / 총괄
- (Module) 스컬핑/픽셀유동화 모듈 구현 + Callback 연동
- (GUI) GUI 개발
- PPT 총괄 및 작성
- 최종발표

# 역할 분배



**Module  
&Liquify**

안준혁

## 프로그램 기획

- 기획안 제시

## Module (Test Tool)

- Mesh 회전, 여러 개 Mesh 동시에 회전
- obj 생성
- obj 여러 개 동시에 로딩 (기즈모 사용)

## Sculpting, Liquify (Sculpting Tool)

- (알고리즘) 픽셀유동화 알고리즘 연구  
(Arap/Mesh Deformation 조사).
- (Module) 스컬핑/픽셀유동화 모듈 구현 + Callback 연동
- (GUI) 메뉴 바 개발 + GUI 정리
- PPT 작성

# 역할 분배



조예진

## 프로그램 기획

- 기획안 제시
- 프로젝트 방향성, 기능 세부사항, 세부일정 수립 (총괄)

## GUI (Test Tool)

- 메인 편집 바 개발 및 연동  
(Zoom/Translate/Rotation 연동)
- 메뉴 바 개발 (Obj 프로젝트 불러오기) 및 연동

## Sculpting, Liquify (Sculpting Tool)

- (알고리즘) 픽셀유동화 알고리즘 연구(Arap 조사), flowchart
- (Module) 스컬핑/픽셀유동화 모듈 구현 + Callback 연동
- (GUI) 메뉴 바 개발 + GUI 정리
- PPT 작성

# 역할 분배

**GUI  
& Liquify**

안선영

## 프로그램 기획

- 기획안 제시

## GUI (Test Tool)

- 메인 편집 바 개발
- 메뉴 바 개발 및 연동  
(Tools, Field, Rendering, Select, Edit, File)
- 사이드 바 개발 및 연동

## Sculpting, Liquify (Sculpting Tool)

- (알고리즘) 픽셀유동화 알고리즘 연구
- (Module) 스컬핑/픽셀유동화 모듈 구현 + Callback 연동
- (GUI) 메뉴 바, 사이드 바 개발 + 전체 연동 및 정리
- PPT 작성



**Thank you**