컴퓨터애니메이션실습(CAL)

HW5_Mesh Component Selection

Self-scoring table

	P1	P2	P3	E1	E2	E3	Total
Score	**1**	**1**	**1**	**1**	**1**	**1**	**6**

2018707068 김경환

KwangWoon University

Code Analysis:

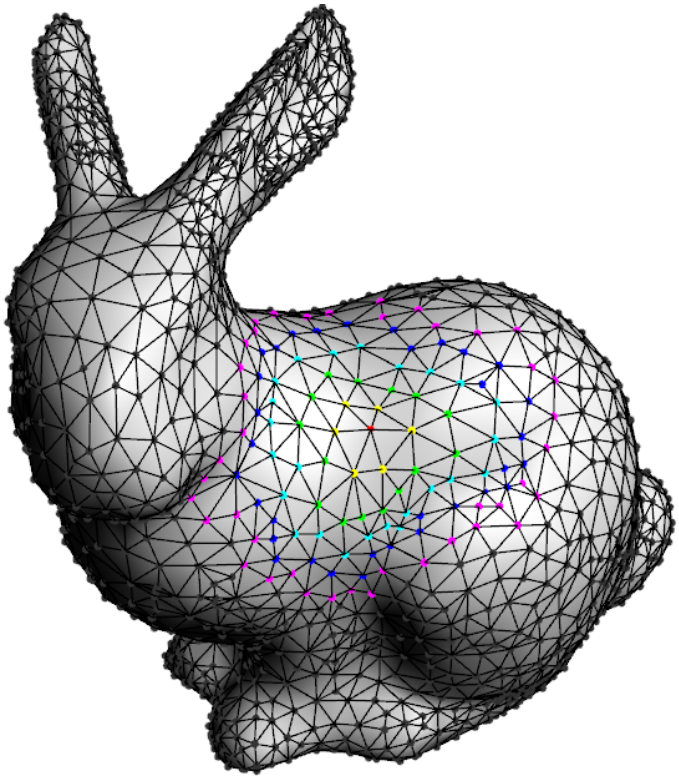
먼저 사용되는 자료구조에 대해 알아보겠다.
Eijf는 vertex i에서 vertex j로 향하는 edge를 생각했을 때 해당 edge의 오른쪽에 위치한 face를 저장한다.
Eijf[i] = {j, f}와 같은 형식으로 저장되며 unordered_map에 따라 Eijf[i].first = j, Eijf[j].second = f이다.
E는 edge e를 저장해놓은 자료구조로써 start vertex와 end vertex의 pair가 저장된다.
E[e] = {start, end}와 같은 형식으로 저장되며 E[e].first = start, E[e].second = end이다.
Ev는 vertex v를 공유하는 edge를 저장해놓은 자료구조로써 edge e가 저장된다.
Ev[v][sequence j] = e와 같은 형식으로 저장되며 j는 [0~vertex v를 공유하는 edge의 개수]이다.
Fv는 vertex v를 공유하는 face를 저장해놓은 자료구조로써 face f가 저장된다.
Fv[v][j] = f와 같은 형식으로 저장되며 j는 [0~vertex v를 공유하는 face의 개수]이다.
Dv, De, Df는 선택된 face, edge, vertex id의 distance를 저장한다. Dvef[id] = distance

findNeighbor(Vertices, Edges, Faces) 모두 Bread First Search를 통해 얻어진다. 여기서 vertex는 adjacent한 vertex를 구할 때 해당 vertex를 공유하는 edge를 통해 neighbor vertex를 구한다. edge는 adjacent한 edge를 구할 때 해당 edge를 이루는 두 vertex를 공유하는 edge를 구해 neighbor edge를 구한다. Vertex, edge 모두 vertex 혹은 edge로 한 가지 경우로만 adjacent한 neighbor를 구한다. 하지만, face의 경우 특이하게 vertex를 통해 adjacent한 face, edge를 통해 adjacent한 face로 두 가지 경우가 존재한다.

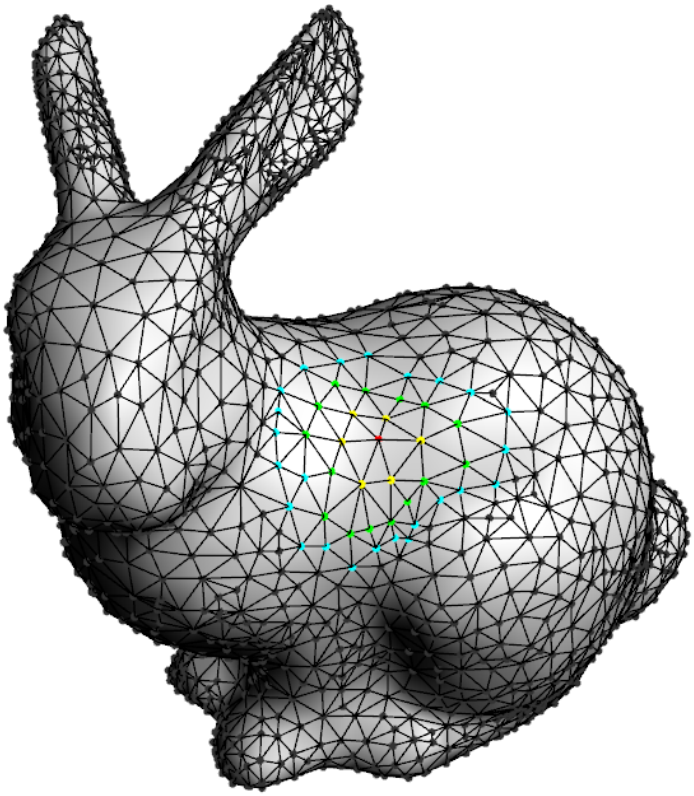
Vertex, Edge, Face를 그리는 방법은 모든 vertex, edge, face를 그릴 때 선택된 component와의 distance가 초기화 값인 -1이 아닌 값일 경우 distance에 따른 색깔로 그림을 그리고, 선택되지 않은 component를 위해 다시 색깔을 돌려놓는 방식으로 그려진다.
그리고 drawing 도중에 picking을 위한 entry의 category, name이 저장된다. 이는 glPushName(enum PickMode (FACE, EDGE, VERTEX))과 glPushName(-1)을 통해 category를 설정한다. 여기서 glPushName(-1)은 해당 category에 최소한 한 개의 entry를 넣기 위해 수행한다. Entry의 name은 glLoadeName()을 통해 저장되고 마지막에 glPopName()을 통해 현재 선택된 이름 스택에서 가장 위에 있는 이름을 제거하고, 이전 이름 스택으로 돌아간다. 그리고 만약 현재 선택된 이름 스택이 루트 스택인 경우, 이 함수는 아무런 동작을 수행하지 않는다.

Component를 선택하는 과정은 먼저 기존에 선택된 Component를 pickedNeighbor(Faces, Edges, Vertices)를 통해 초기화 값 -1로 돌려놓는 과정을 거친다. 그리고 지난학기 컴퓨터 그래픽스 수행한 selectObject()을 통해 선택된 Component의 name을 얻어내고 해당 name을 통해 Component distance를 수정, pickNeighbor(Faces, Edges, Vertices)에 추가를 진행한다. 마지막으로 nRing >= 1일 경우 findNeighbor(Faces, Edges, Vertices)를 수행해 Pick된 component와 adjacent한 component를 갱신한다.

Practice 01. Picking neighbors of a vertex:



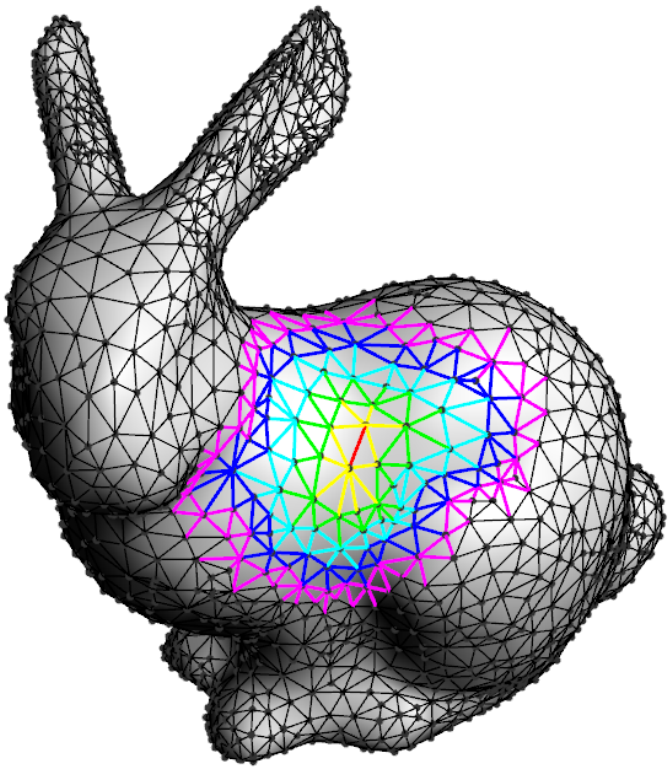
Vertex를 Picking하여 5-Ring까지 표현한 모습



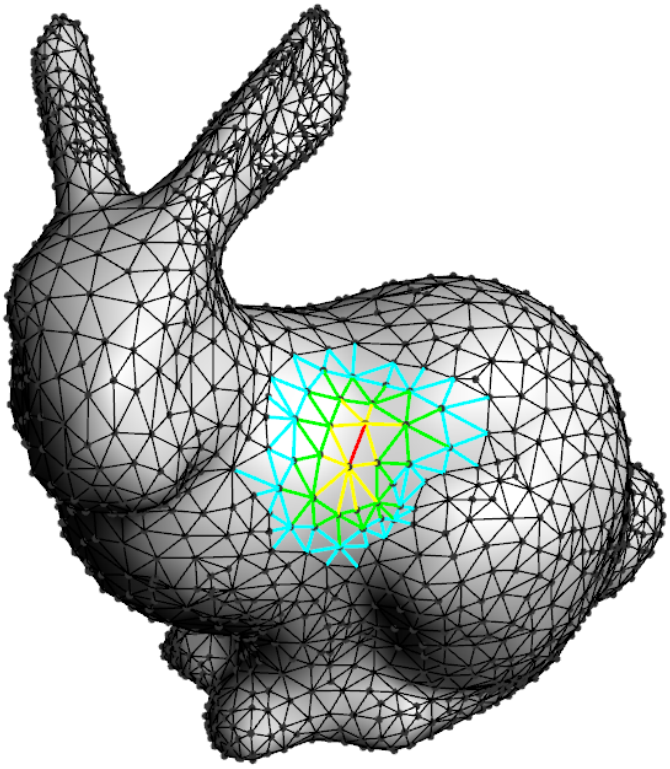
Vertex를 Picking하여 3-Ring까지 표현한 모습

Ring은 Pick된 Component와의 distance에 따라 (0: Red, 1: Yellow, 2: Green, 3: Cyan, 4: Blue, 5: Magenta)

Practice 02. Picking neighbors of an edge:



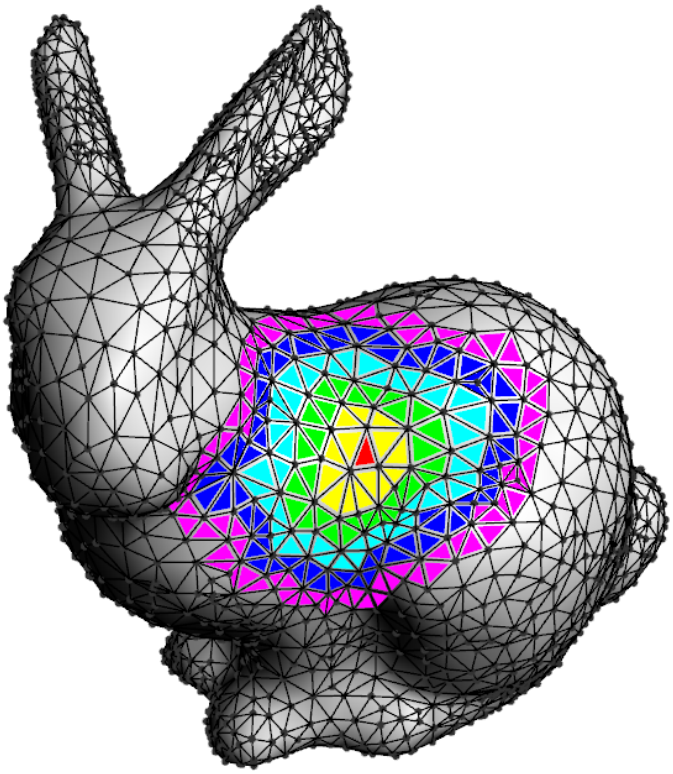
Edge를 Picking하여 5-Ring까지 표현한 모습



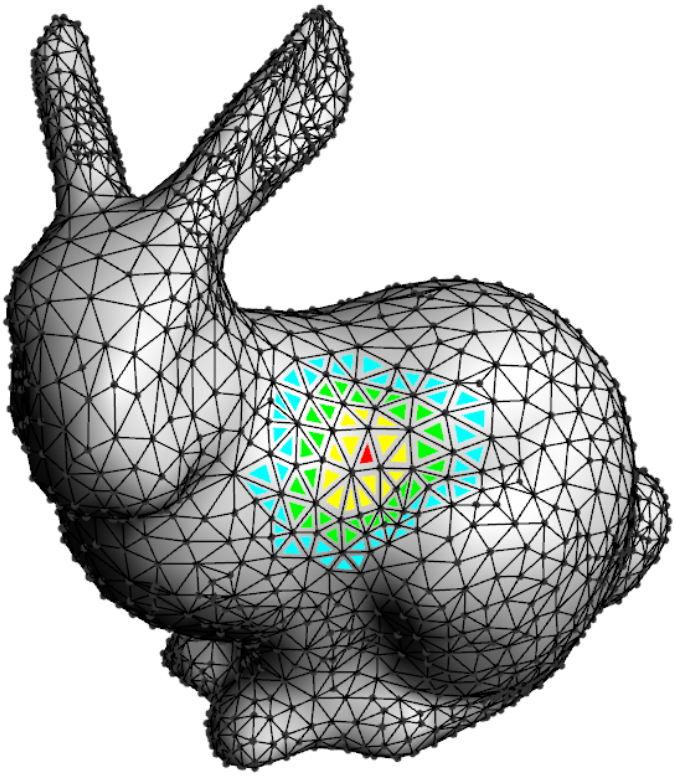
Edge를 Picking하여 3-Ring까지 표현한 모습

Ring은 Pick된 Component와의 distance에 따라 (0: Red, 1: Yellow, 2: Green, 3: Cyan, 4: Blue, 5: Magenta)

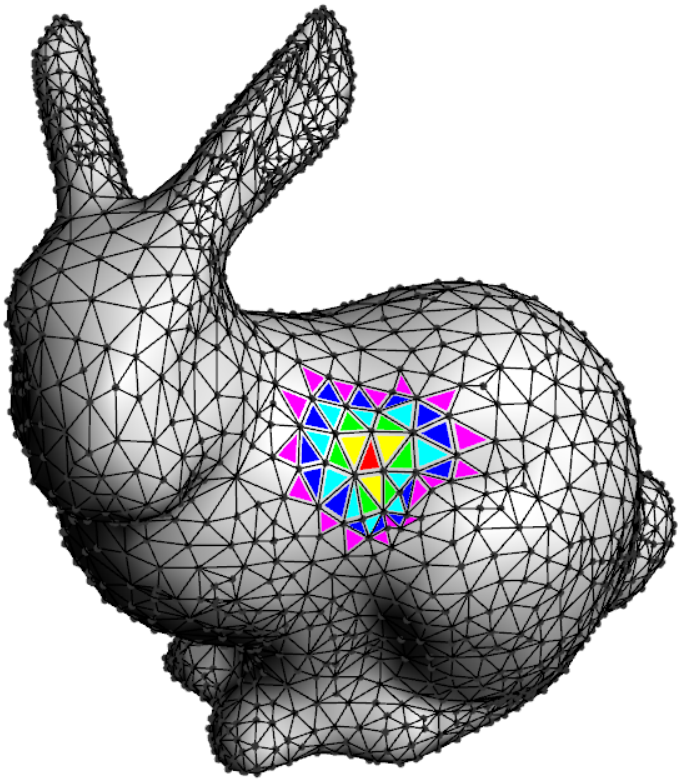
Practice 03. Picking neighbors of a face:



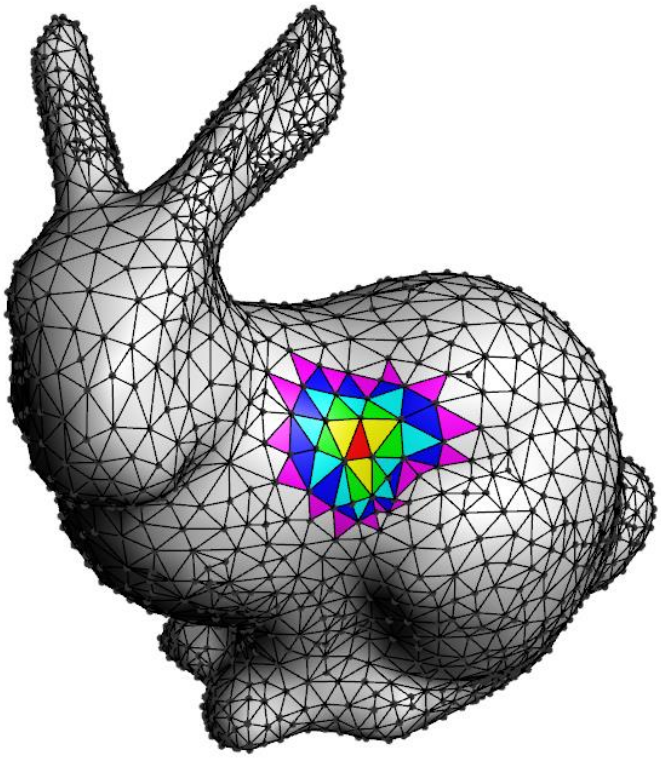
Face를 Picking, vertex Adjacent 5-Ring



Face를 Picking, vertex Adjacent 3-Ring, gap 축소



Face를 Picking, edge Adjacent 5-Ring



Face를 Picking, vertex Adjacent 5-Ring, gap 제거

Ring은 Pick된 Component와의 distance에 따라 (0: Red, 1: Yellow, 2: Green, 3: Cyan, 4: Blue, 5: Magenta)

Exercise Code Analysis:

기존 Select() 함수에서 n-Ring 기능이 빠져 select()는 기존에 선택된 component를 초기화하는 역할만 한다. 그리고 selectobject를 호출하고 해당 함수에서 findNearestHits을 수행하는데 여기서 hit된 모든 componentname에 대하여 선택됐음을 나타내는 component distance인 Dvef를 componentName으로 접근해 값을 0으로 설정한다. 그리고 pickedNeighbor(Vertices, Edges, Faces)에도 해당 ComponentName과 distance 0을 추가한다. 이를 통해 선택된 component들은 render()에서 다른 색상으로 그려질 것이다.

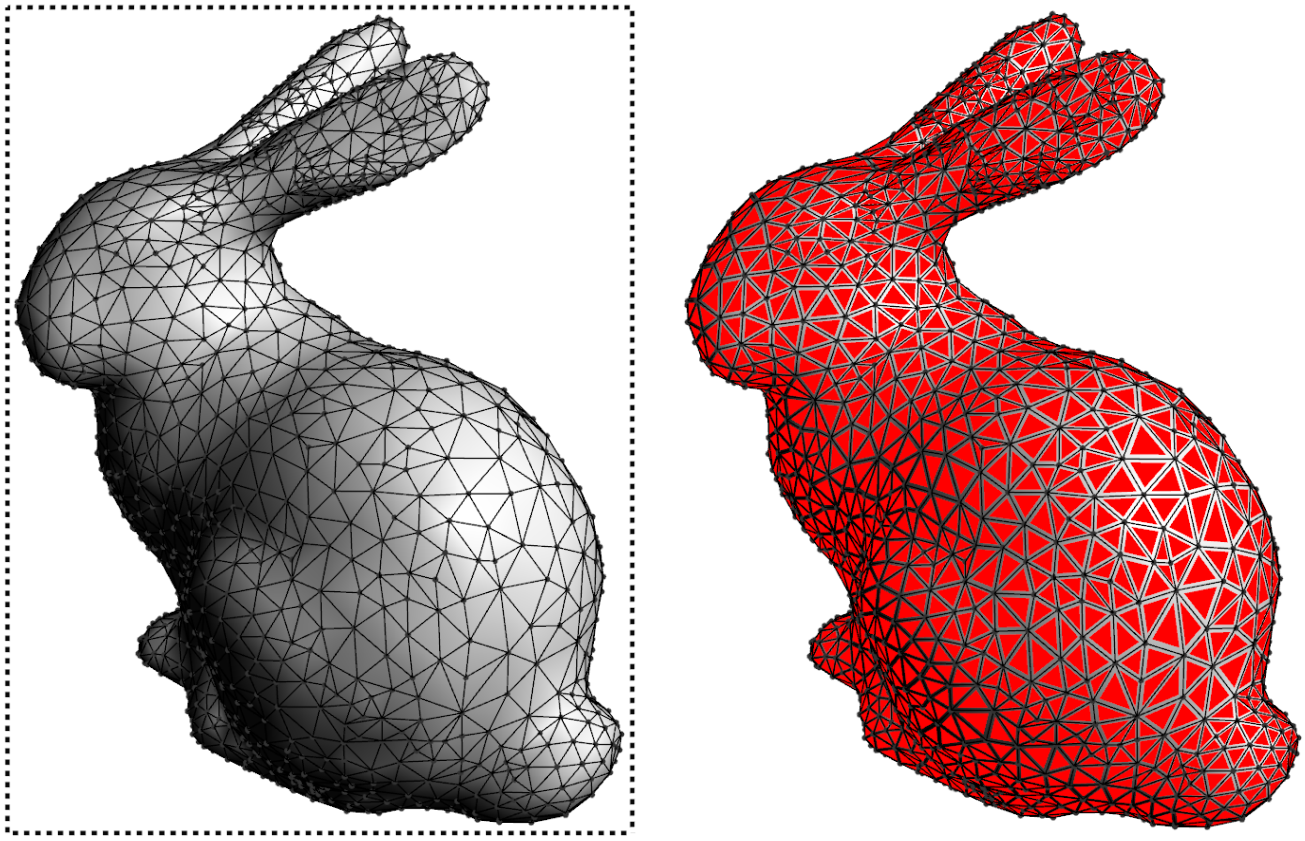
click and drag로 stippling rectangle을 렌더링하기 위해서는 rectangle의 좌측상단과 우측하단의 좌표가 필요하다. 그리하여 mouseButton()에서 LEFT를 Press했을 때 dragging 모드로 바꾸고 glfwGetCusorPos를 사용하여 두 좌표를 모두 Cusor의 좌표로 저장한다. 그리고 dragging한 상태로 mouse를 move할 때는 rectangle의 모양이 바뀌어야 함으로 mouseMove()에서 dragging 모드일 때 우측하단의 좌표를 현재 Cusor의 좌표로 갱신한다. 그리 고 mouse의 Left를 Release하여 rectangle의 크기를 확정지을 때는 complete mode로 바꾸고 현재 Cusor 의 좌표를 rectangle의 우측하단의 좌표로 저장한다.
이렇게 rectangle의 좌표를 얻었으면 이를 렌더링하기 위해서 좌표계를 조정할 필요가 있다. mouse의 좌표는 window coordinate이기 때문에 bunny model이 perspective projection되어 있는 환경에 올바르게 렌더링할 수 없기 때문이다. 그리하여 render()에서 drawRectangle() 함수를 호출한다. 이때 rectangle은 mouse를 dragging할 때만 렌더링 되어야 함으로 if문을 사용한다. drawRectangle()은 위에서 말한 좌표계의 조정이 이루어지기 전에 screen 좌표를 world 좌표로 바꾸는 screen2world()를 사용한다. 그렇게 얻은 world coordinate를 렌더링하기 위해 우선 기존 projection matrix를 backup하고 단위행렬로 초기화한다. 이는 rectangle을 렌더링하고 다시 이전 projection matrix로 돌려놓기 위함이다. 그리고 perspectiveView에 false를 저장하고 setupProjectionMatrix()를 호출하여 orthogonal projection으로 바꾼다. 또한, modelview matrix의 경우에도 현재 2D world coordinate를 렌더링할 때는 영향을 주지 않아야 함으로 단위행렬로 초기화한다. 이제 이전에 얻은 world coordinate를 렌더링하고 다시 backup 해둔 projection matrix를 돌려놓는다. 마지막으로 기존에는 perspective projection이었기 때문에 perspectiveView를 true로 바꿔놓는다. 이렇게 perspectiveView를 true로 바꿔놓아야 하는 이유는 selectObjects()에서 setupProjectionMatrix()을 호출하는데 perspectiveView이 false로 되어 있으면 orthogonal로 올바른 select가 되지 않는다. 마지막으로 여기서 modelview matrix는 단위행렬로 초기화하고 기존 행렬로 다시 바꿔놓지 않았는데 이는 render()에서 modelview matrix를 계속해서 viewport에 맞춰 다시 설정하기 때문이다.

practice에서 작성된 selectObjects()와 findNearestHits()에서 사용되는 selectBuffer를 보면 selectObjects()에서는 [64*3]의 크기로 선언되었지만 findNearestHits()에서 매개변수로 전달받을 때는 [64]의 크기로 전달받아 이 부분에서 헤맸었다. 또힌 findNearestHits()에서 [64] 크기의 selectBuffer를 전달받아 64보다 더 큰 index로 해당 배열을 접근하기도 하였기에 이해가 안 갔었는데, 찾아보니 매개변수로 배열을 받을 때 사용되는 배열의 크기는 의미가 없다고 한다. 어짜피 해당 배열은 포인터로 전달되기 때문이고 이로 인해서 64보다 더 큰 index로도 접근할 수 있었던 것이다.

Click-and-drag 안의 component를 선택할 때 mouse를 release하고도 select의 반응이 느렸던 문제가 있었는데 이는 diagnosis가 true로 되어있어 select된 component의 정보를 모두 출력하느라 반영되지 않았던

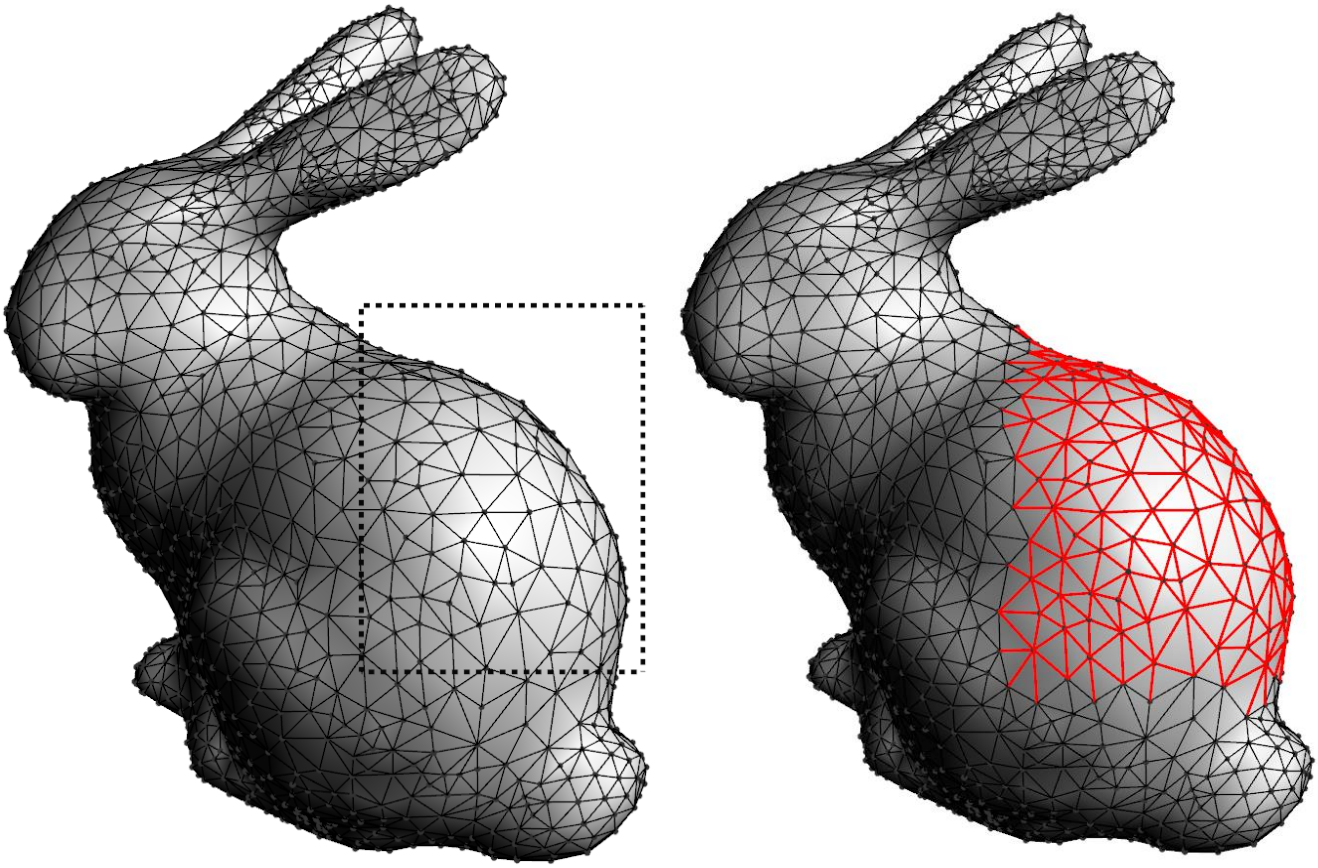
것이었다. 이는 diagnosis를 false로 바꾸면 완벽하게 해결된다.

Exercise 01. Select multiple vertices by click-and-drag:



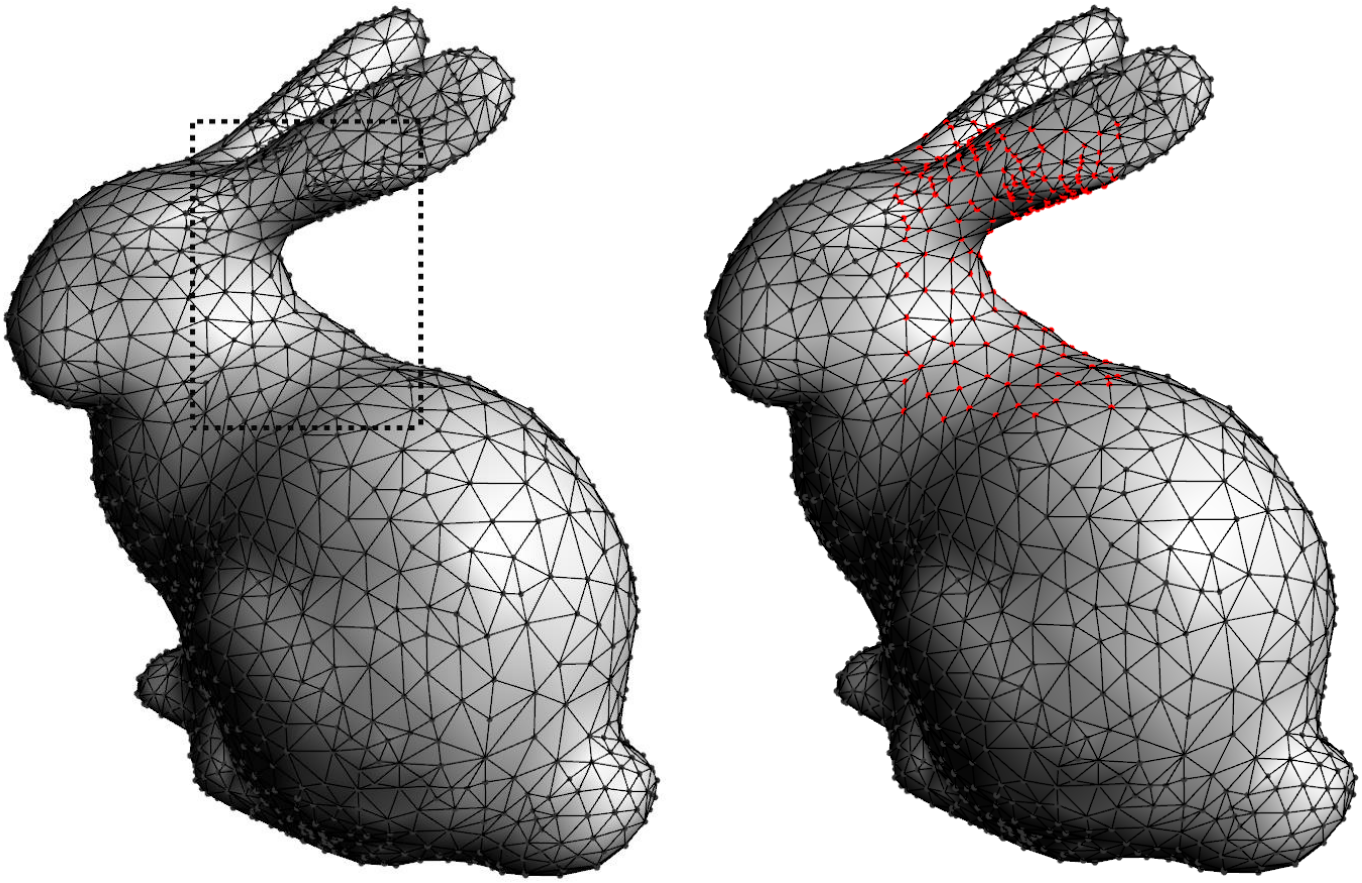
Click-and-drag로 vertex를 multiple select한 모습

Exercise 02. Select multiple edges by click-and-drag:



Click-and-drag로 edge를 multiple select한 모습

Exercise 03. Select multiple faces by click-and-drag:



Click-and-drag로 face를 multiple select한 모습