2023학년도 1학기

컴퓨터애니메이션실습

컴퓨터 애니메이션

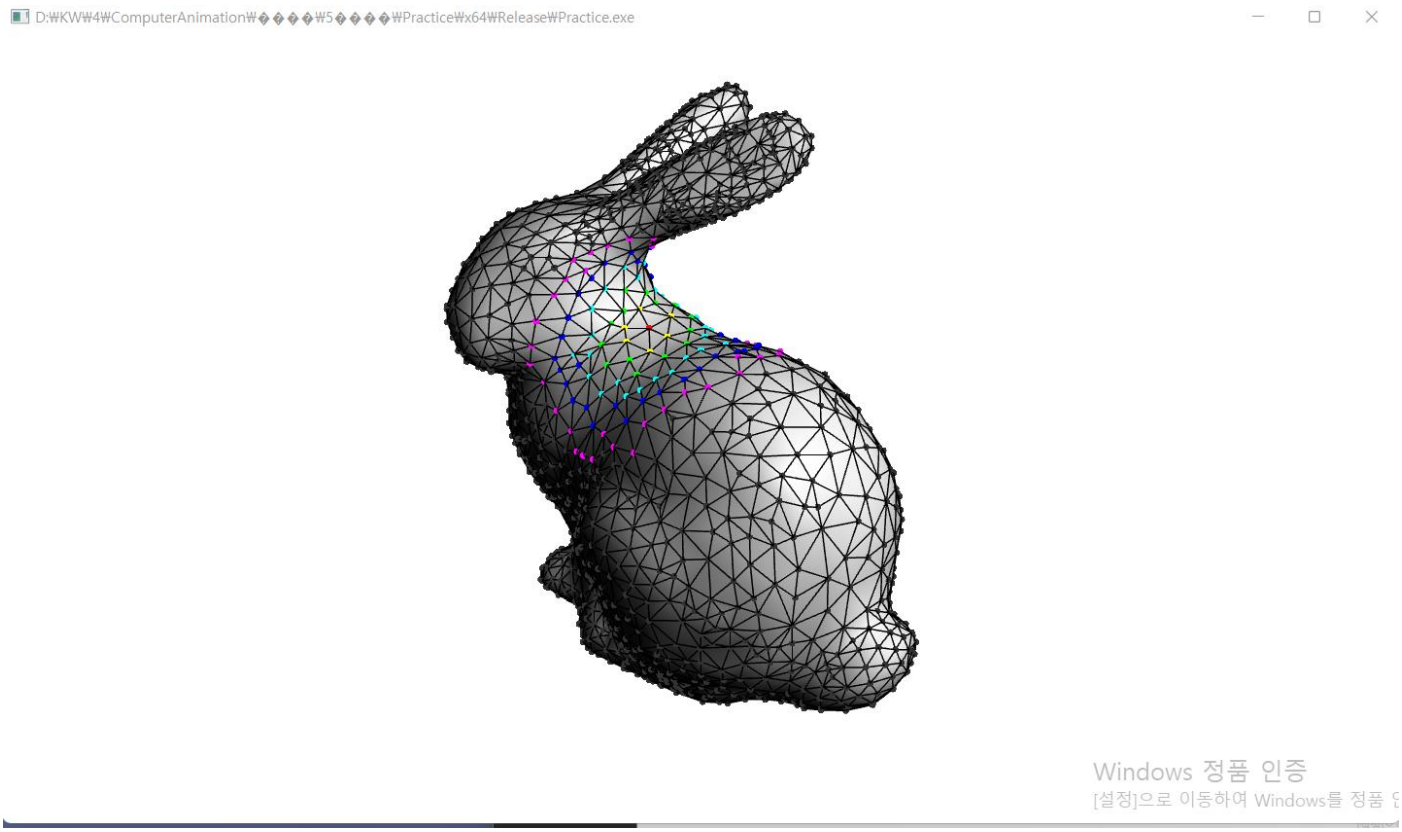
실습 보고서

Self-Scoring Table

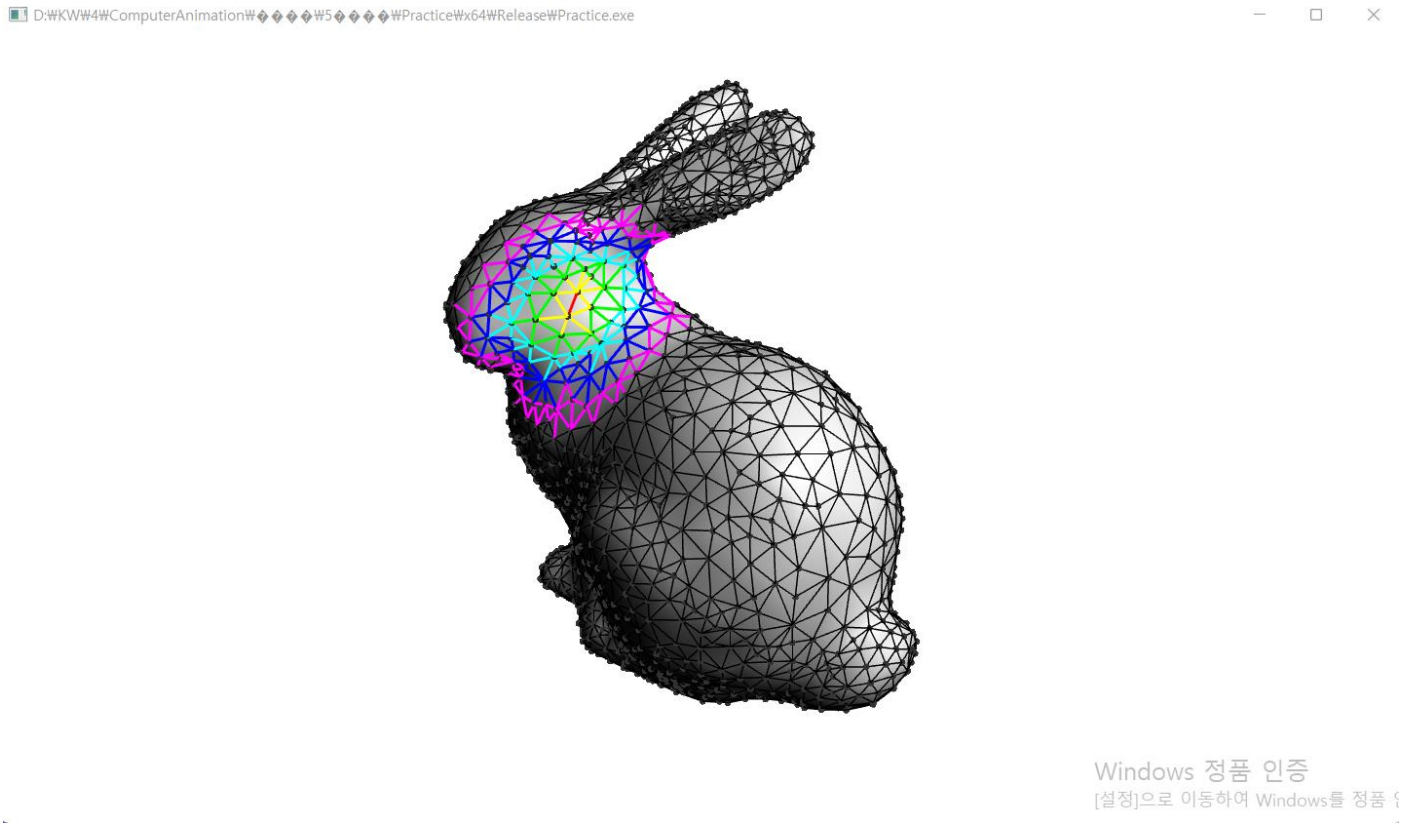
	P1	P2	P3	E1	E2	E3	Total
Score	1	1	1	1	1	1	6

정보융합학부 2018204058 김민교

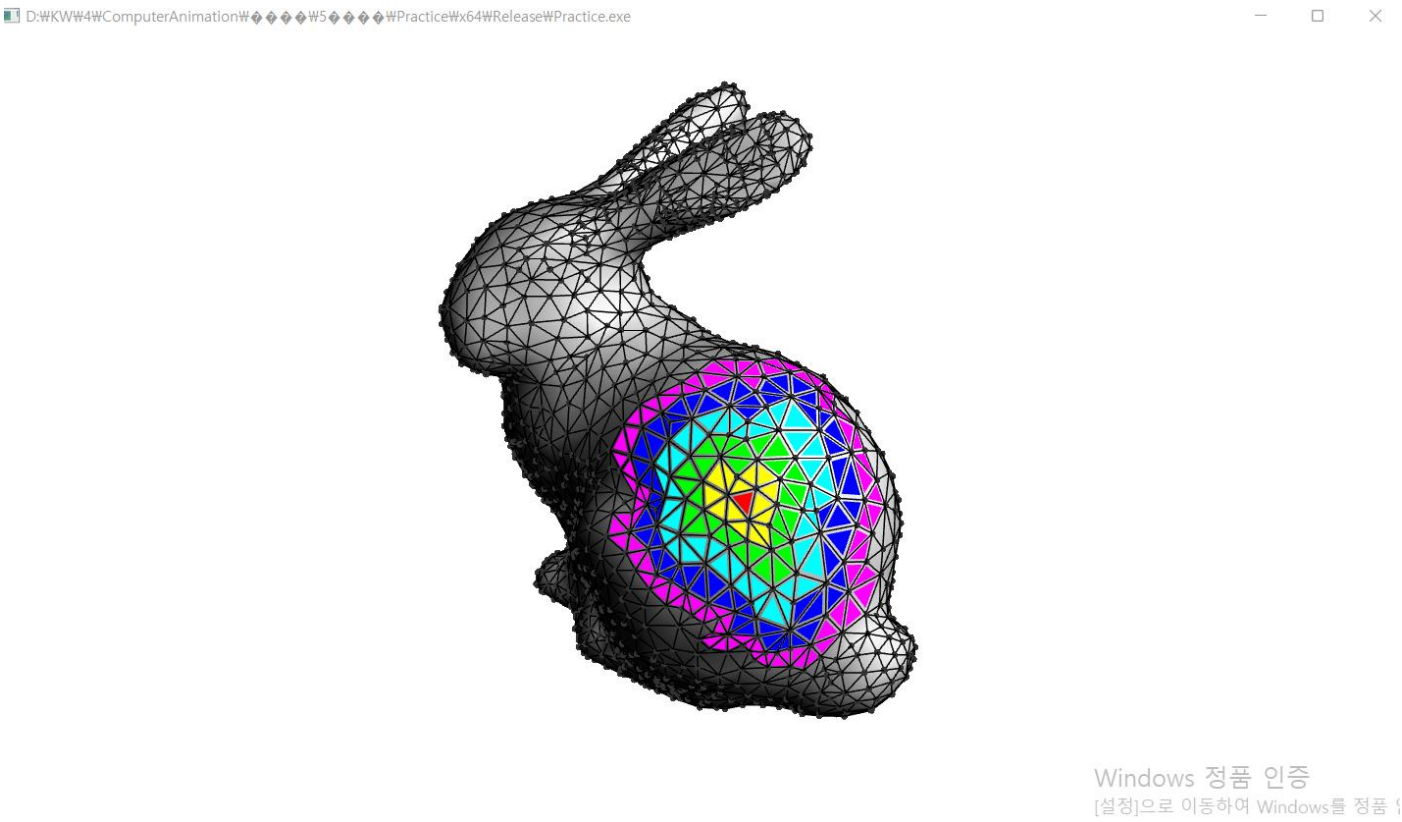
P1 - Picking neighbors of a vertex



P2 - Picking neighbors of an edge



P3 - Picking neighbors of a face



- **코드 설명**

SELECTION 모드

객체를 선택하기 위해서 렌더링 모드 변경을 해야한다.

glRenderMode(GL_SELECTION); // 선택모드로 변경

glRenderMode(GL_RENDER): // 드로잉 모드로 변경

glInitNames();

Name stack을 초기화한다. NameStack이란? 마우스를 이용해서 물체를 선택했을 때, 어떤 물체인지 어떻게 알까? 바로 이름을 붙여주는 것이다. 물체의 이름을 숫자로 정해준다. NameStack이란 물체의 이름을 저장할 공간이다.

glPushName()

이 코드 이후로부터 glPopName 이전까지 그려지는 모든 물체에 대해 지정된 이름을 붙인다. 함수의 인수가 이름이 된다.

```
// Picking
glPushName(VERTEX); // Push the category name in the stack
glPushName(-1);     // Vertex id in the category
```

selectBuffer : 물체가 선택되면 선택된 물체의 이름이 selectBuffer에 저장된다.

```
845        glSelectBuffer(64 * 5, selectBuffer);
```

selectBuffer로 사용될 버퍼의 크기와, 그 버퍼로 사용될 메모리 영역을 잡아준다

물체를 클릭하면 selectbuffer 4번째 셀에 클릭한 물체의 이름을 저장한다. 다음 물체를 클릭하면 8번째, 12번째… 이렇게 간격이 4가 된다.

gluPickMatrix() : 선택 영역을 결정한다.

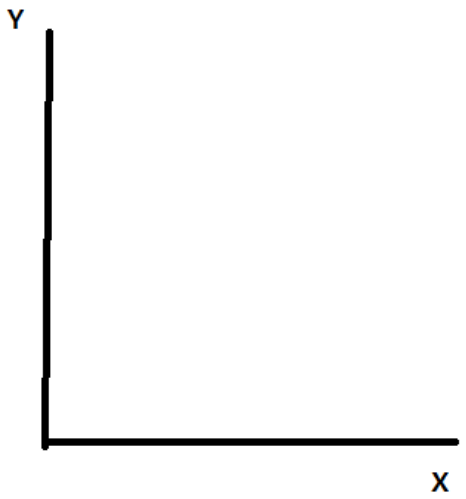
첫번째 인자 : 선택 영역 가운데 점의 X좌표

두번째 인자 : 선택 영역 가운데 점의 Y좌표

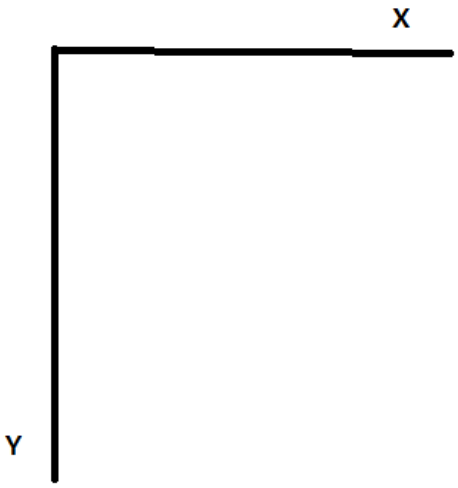
세번째 , 네번째, 주변으로 얼마만큼 위치에 있는 물체도 선택할지.

```
858        // a small region of the viewport
859        gluPickMatrix(x, viewport[3] - y, delX, delY, viewport); // y screen to viewport
```

openGL의 스크린 스페이스 좌표계와 윈도우 스크린 스페이스 좌표계는 달라서 변환해주어야 한다.viewport[3]은 클라이언트 영역의 높이 값이다. viewport[3] - 윈도우 y좌표를 한다면 opengl의 스크린스페이스에서의 y좌표를 구할 수 있다.



GL Screen 좌표계

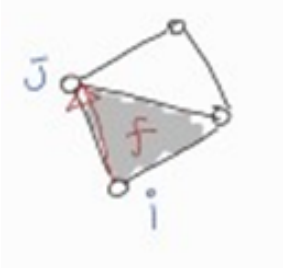


Window Screen 좌표계

N-ring을 구현하기 위한 자료구조

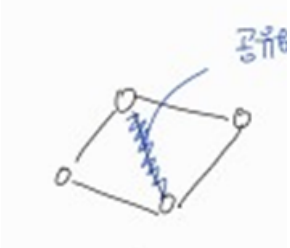
vector<unordered_map<int, int>> Eijf : E[i] = {j, f} 형태.

vertex i에서 vertex j로 가는 엣지의 오른쪽 면을 저장합니다.



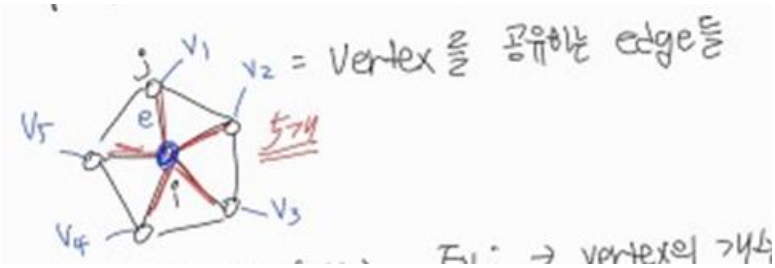
vector<pair<int,int>> E: E[i] = {start, end} 형태

엣지 i를 구성하는 vertex 두개를 저장합니다.



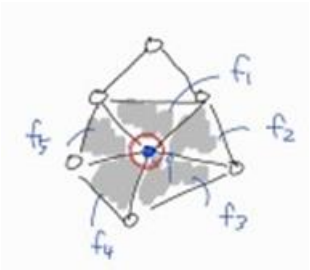
vector<vector<int>> Ev : Ev[v][j] = e 형태

한 vertex의 인접한 엣지들을 찾습니다. E[v]에는 여러 엣지들이 저장되어 있습니다.

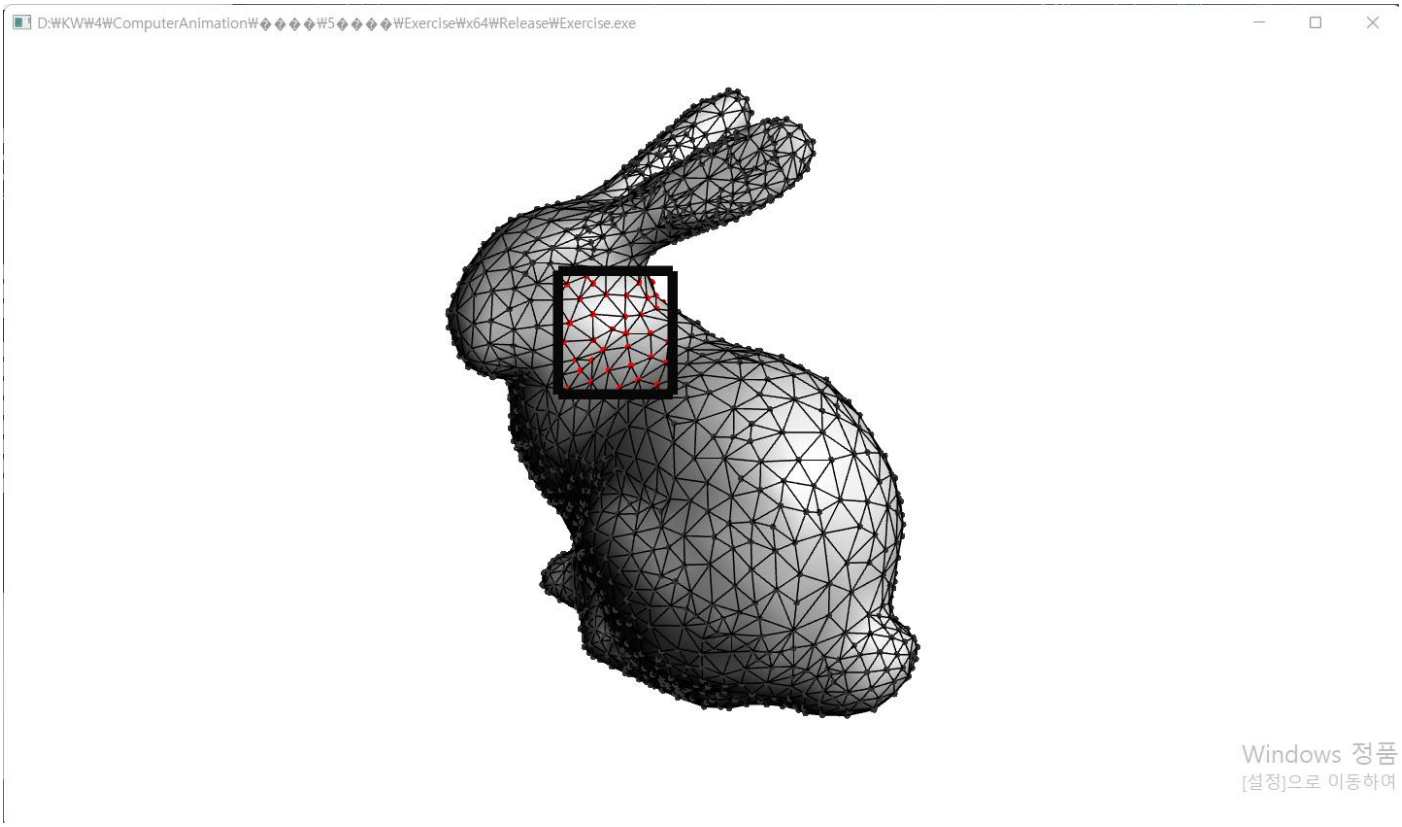


vector<vector<int>>Fv : Fv[v][j] = f 형태

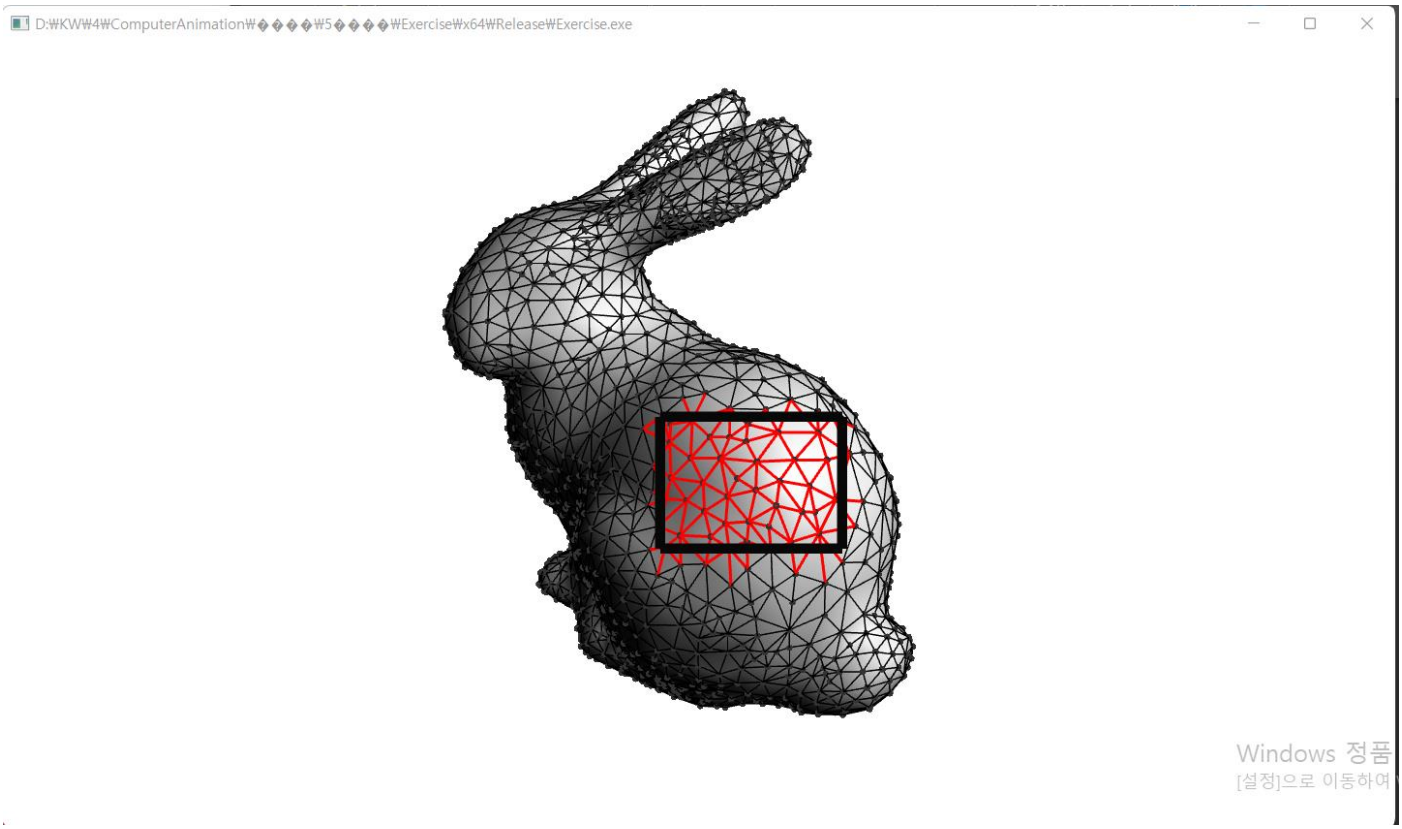
어느 한 vertex에 인접한 face들을 저장합니다. Fv[v]에는 여러 face들이 저장되어 있습니다.



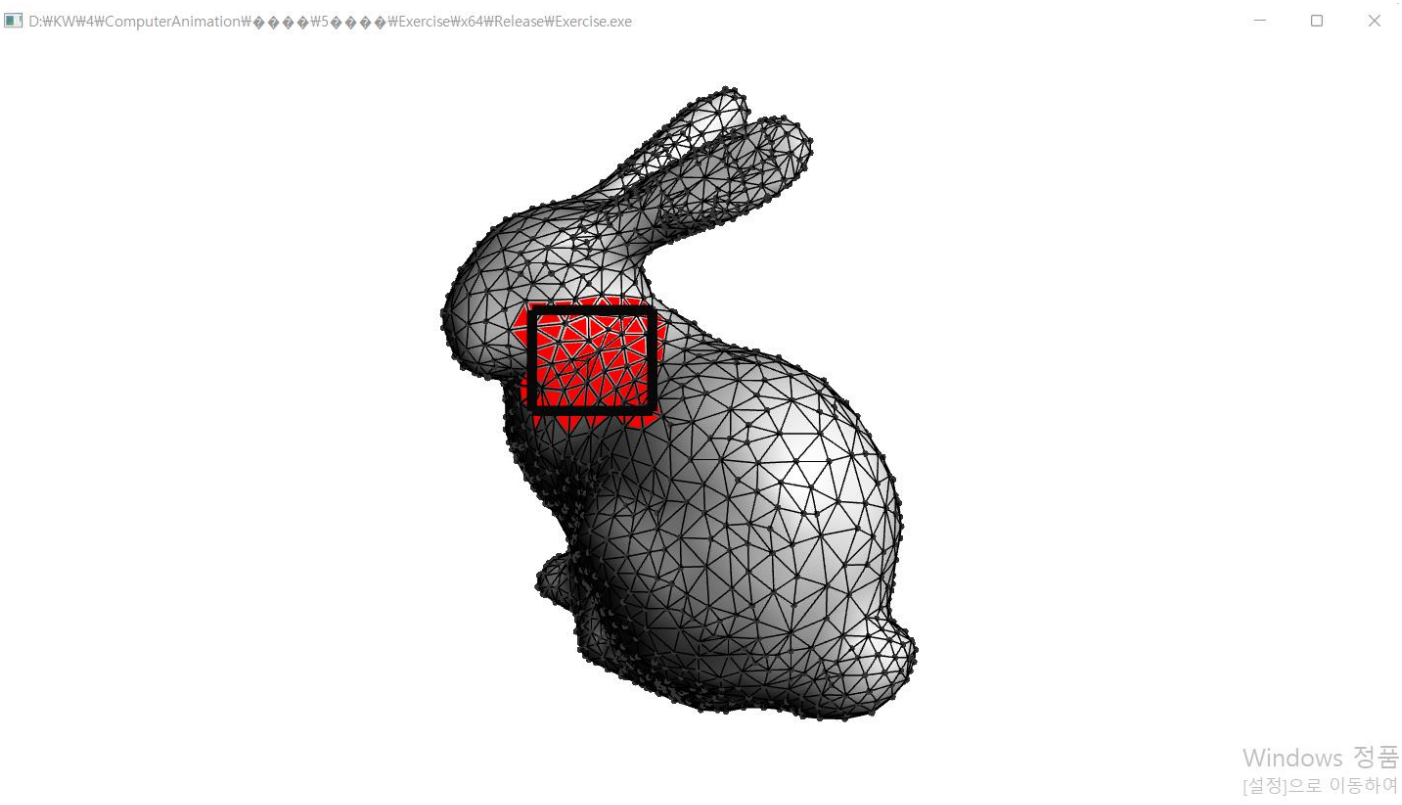
E1 - Select multiple vertices by click - and - drag



E2 - Select multiple edges by click - and - drag



E3 - Select multiple faces by click - and - drag



- 코드 설명

```
992  void mouseButton(GLFWwindow* window, int button, int action, int mods
993  {
994      if (action == GLFW_PRESS && button == GLFW_MOUSE_BUTTON_LEFT)
995      {
996          drag = false;
997          // Mouse cursor position in the screen coordinate
998          double xs, ys;
999          glfwGetCursorPos(window, &xs, &ys);
1000
1001         // Mouse cursor position in the framebuffer coordinate
1002         preX = xs * dpiScaling;
1003         preY = ys * dpiScaling;
1004     }
1005     else if(action == GLFW_RELEASE)
1006     {
1007         select(window, newX, newY);
1008     }
1009 }
```

mouse button을 처음 누르면 drag를 false로 초기화 해준다. 처음 클릭한 스크린 스페이스에서의 x, y좌표를 저장해둔다.

mouse button을 떼었을 때, drag를 전부 한 것이니 이제 select를 해준다. select의 인수로 NewX와 newY값이 있는데, 이는 mouse move할 때 바뀌는 값이다.

```
1011 void mouseMove(GLFWwindow* window, double xs, double ys)
1012 {
1013     if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT) == GLFW_RELEASE) return;
1014     drag = true;
1015
1016     // Mouse cursor position in the framebuffer coordinate
1017     double x = xs * dpiScaling;
1018     double y = ys * dpiScaling;
1019
1020     newX = x;
1021     newY = y;
1022 }
```

mouse move를 하면 drag를 하고 있다는 뜻이니, drag값을 true로 변경해준다. mouse가 이동하면서 x, y 값을 계속 갱신해준다.

```
719     // Draw rectangle if dragging
720     if (drag)
721     {
722         drawDragRectangle();
723     }
724
```

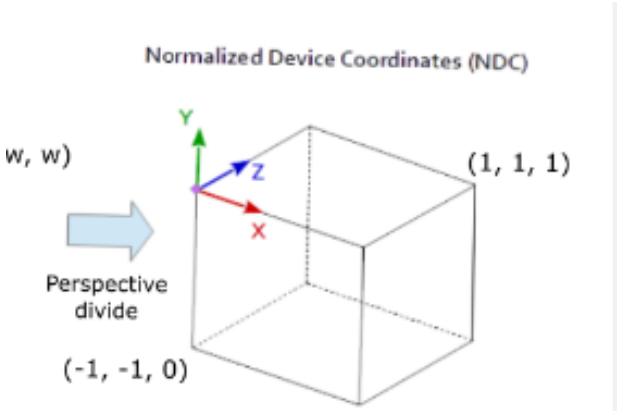
Render 함수에서 drag값이 true일 때만 사각형을 그려준다.

```
746  void drawDragRectangle()
747  {
748      GLdouble X1_ws, Y1_ws,Z1_ws , X2_ws, Y2_ws, Z2_ws, X3_ws, Y3_ws, Z3_ws, X4_ws, Y4_ws, Z4_ws;
749
750      // screen coordinate to world coordinate.
751      // x1, y1,z1 , x2, y2,z3, x3, y3,z3, x4, y4,z4 are corner of rectangle in world coordinate.
752      unProject(preX,preY,&X1_ws,&Y1_ws,&Z1_ws);
753      unProject(newX, preY,&X2_ws, &Y2_ws, &Z2_ws);
754      unProject(preX, newY, &X3_ws, &Y3_ws, &Z3_ws);
755      unProject(newX, newY, &X4_ws, &Y4_ws, &Z4_ws);
756
757      glColor3f(0.0, 0.0, 1.0);
758      glLineWidth(11);
759      glBegin(GL_LINE_LOOP);
760      glVertex3f(X1_ws, Y1_ws,Z1_ws);
761      glVertex3f(X2_ws, Y2_ws,Z2_ws);
762      glVertex3f(X4_ws, Y4_ws,Z4_ws);
763      glVertex3f(X3_ws, Y3_ws,Z3_ws);
764      glEnd();
765  }
```

drawDragRectangle 함수에서 그려질 사각형의 네 꼭짓점 좌표를 월드 스페이스로 변환한다. 변환한 월드상의 x, y, z로 LineLoop를 통해 라인을 그리면서 사각형을 그린다.

```
727  void unProject(double x, double y, GLdouble *wx, GLdouble *wy, GLdouble *wz)
728  {
729      GLdouble projection[16];
730      GLdouble modelView[16];
731      GLint viewPort[4];
732      glGetDoublev(GL_PROJECTION_MATRIX, projection);
733      glGetDoublev(GL_MODELVIEW_MATRIX, modelView);
734      glGetIntegerv(GL_VIEWPORT, viewPort);
735
736      GLfloat zCursor, winX, winY;
737      // window coordinate X, Y, Z change to openGL screen coordinates
738      winX = (float)x;
739      winY = (float)viewPort[3] - (float)y;
740
741      if (gluUnProject(winX, winY, 0, modelView, projection, viewPort, wx, wy, wz) == GLU_FALSE) {
742          printf("failed to unproject\n");
743      }
744  }
```

먼저 윈도우 스크린 스페이스 좌표를 OpenGL 스크린 스페이스 좌표로 변경해준다. 그 다음 gluUnProject 함수를 통해 스크린 스페이스값을 월드 스페이스 값으로 변환한다. 이 때, winZ값에 0이 들어갔는데, 그 이유는 사각형의 z depth를 가장 낮게 함으로써 렌더링 될 때 제일 앞에 보이게 하기 위함이다.



z가 0일 때 near이고 1일 때 far라서 가장 가까운 0값을 주었다.

```
870             if (categoryName == pickMode)
871             {
872                 int     componentName = selectBuffer[index + 4]; // 각 Category에서 몇 번째인지
873 
874                 names.push_back(componentName);
875 
876                 //if (z1 < nearest) { nearest = z1; name = componentName; }
877 
```

picking된 것을 고를 때, practice에서는 가장 가까운 것을 골랐지만, 여기서는 전부 가져와야 한다. 그래서 벡터 자료구조를 만들고 전부 저장해놓는다.

```
968         switch (pickMode)
969         {
970         case VERTEX:
971             for (int i = 0; i < Multiplepicked.size(); i++)
972             {
973                 Dv[Multiplepicked[i]] = 0;
974             }
975             break;
976         case EDGE:
977             for (int i = 0; i < Multiplepicked.size(); i++)
978             {
979                 De[Multiplepicked[i]] = 0;
980             }
981             break;
982         case FACE:
983             for (int i = 0; i < Multiplepicked.size(); i++)
984             {
```

selectObjects 함수를 통해 Multiplepicked 벡터에 선택된 객체들이 담긴다. 반복문을 통해 선택된 객체들을 하나씩 조회하면서 색칠한다.

```
949     // Rest the previous selections
950     if (Multiplepicked.size() != 0)
951     {
952         for (int i = 0; i < Multiplepicked.size(); i++
953         {
954             if(nFaces> Multiplepicked[i])
955                 Df[Multiplepicked[i]]= -1;
956             if (nVertices> Multiplepicked[i])
957                 Dv[Multiplepicked[i]] = -1;
958             if (nEdges > Multiplepicked[i])
959                 De[Multiplepicked[i]] = -1;
960         }
961         Multiplepicked.clear();
962     }
```

색칠한 것을 초기화 할 때에는 먼저 MultiplePicked 벡터가 비어있는지 확인한다. 비어있지 않으면 이제 색칠했던 것들을 지우고 MultiplePikced 벡터 또한 초기화한다.