**BÀI THỰC HÀNH SỐ 2**

Học kỳ 1 - Năm học 2024-2025

**Mục đích:**

- Xây dựng lớp Mesh

- Thực hành về tạo lưới đa giác biểu diễn đối tượng 3 chiều

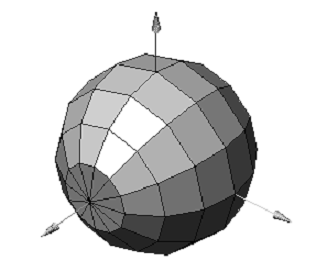
**Các công việc cần thực hiện:**

* Đọc hiểu phần 1 của bài thực hành số 2. Sinh viên có thể đọc thêm file Luoidagiac.pdf đính kèm
* Chạy chương trình demo Lab2.exe
* Tạo project và đưa các tập tin trong InitialCode.zip vào trong project này
* Thêm vào lớp Mesh các hàm tạo một số khối hình học
* Chỉnh sửa chương trình để có được kết quả giống như chương trình demo

**PHẦN 1 : Giới thiệu về lưới đa giác (sinh viên chỉ cần đọc hiểu)**

Trong đồ họa máy tính, để vẽ các đối tượng 3 chiều, người ta thường dùng lưới đa giác để biểu diễn đối tượng 3 chiều. Việc vẽ đối tượng 3 chiều chẳng qua là vẽ tất cả các đa giác trong lưới đa giác. Có thể dùng lưới đa giác biểu diễn chính xác khối đa diện. Trong khi đó các bề mặt cong chỉ có thể dùng lưới đa giác để biểu diễn gần đúng, tuy nhiên khi số lượng đa giác trong lưới đa giác càng nhiều thì lưới đa giác càng gần với bề mặt cong.

Hình sau minh họa quá trình xây dựng lưới đa giác cho một mặt cầu. Chia hình cầu thành từng lát và từng múi. Trong đồ họa, thuật ngữ tương ứng với lát là “stack”, tương ứng với múi là “slice”. Chia mặt cầu thành nSlices múi quanh đường xích đạo và nStacks lát từ cực bắc đến cực nam. Mặt cầu trong hình vẽ có 12 múi và 8 lát. Khi giá trị nSlices và nStacks càng lớn thì lưới đa giác càng giống mặt cầu.



**Hình 1**

Để tạo lưới đa giác, trước tiên phải tính được tọa độ các đỉnh của lưới đa giác, sau đó nối các đỉnh với nhau.

Ta sử dụng hệ tọa độ cầu để tính tọa độ các đỉnh của lưới đa giác. Sử dụng hai biến chạy *u* (tương ứng với các múi) và *v* (tương ứng với các lát), với mỗi cặp giá trị (*u*, *v*) sẽ tính được một đỉnh của lưới đa giác.

Để tạo ra các múi, thì giá trị *u* của nSlices múi phải nằm giữa 0 và 2π (tức là giữa 0 và 360 độ). Thông thường các giá trị *u* này cách đều nhau, tức là *ui* = 2π*i*/ nSlices với *i* = 0, 1, …, nSlices – 1.

Đối với các lát, chúng ta phân bố một nửa số lát nằm trên đường xích đạo, nửa còn lại nằm dưới đường xích đạo. Lát trên cùng và lát dưới cùng được tạo bởi các tam giác, tất cả các mặt còn lại đều là tứ giác. Điều này đòi hỏi chúng ta phải định nghĩa (nStacks + 1) giá trị *v*, với *vj* = π/2 – π*j*/nStacks (*j* = 0, 1, …, nStacks).

Trong tập tin supportClass.h định nghĩa thêm hai lớp Point3 và Vector3. Trong tập tin supportClass.cpp hiện thực các phương thức của 2 lớp này:

Trong tập tin Mesh.h và Mesh.cpp định nghĩa và hiện thực lớp Mesh. Lớp Mesh cung cấp phương thức tạo hai đối tượng 3 chiều đơn giản đó là hình lập phương và hình tứ diện. Trong bài thực hành này, sinh viên sẽ bổ sung thêm các phương thức vào lớp Mesh để tạo ra một số đối tượng 3 chiều khác.

*a) Lớp Point3 định nghĩa một điểm trong không gian 3 chiều*

class Point3

{

public:

float x, y, z;

void set(float dx, float dy, float dz)

{ x = dx; y = dy; z = dz;}

void set(Point3& p)

{ x = p.x; y = p.y; z = p.z;}

Point3() { x = y = z = 0;}

Point3(float dx, float dy, float dz)

{ x = dx; y = dy; z = dz;}

};

*b) Lớp Vector3 định nghĩa một vector trong không gian 3 chiều*

class Vector3

{

public:

float x, y, z;

void set(float dx, float dy, float dz)

{ x = dx; y = dy; z = dz;}

void set(Vector3& v)

{ x = v.x; y = v.y; z = v.z;}

void flip()

{ x = -x; y = -y; z = -z;}

void normalize();

Vector3() { x = y = z = 0;}

Vector3(float dx, float dy, float dz)

{ x = dx; y = dy; z = dz;}

Vector3(Vector3& v)

{ x = v.x; y = v.y; z = v.z;}

Vector3 cross(Vector3 b);

float dot(Vector3 b);

};

Các hàm normalize(), cross() và dot() được hiện thực trong tập tin supportClass.cpp. Hãy đọc kỹ hai lớp này và cho biết các hàm normalize(), cross() và dot() thực hiện chức năng gì.

*c) Lớp Mesh:*

class VertexID

{

public:

int vertIndex;

int colorIndex;

};

class Face

{

public:

int nVerts;

VertexID\* vert;

Face()

{

nVerts = 0;

vert = NULL;

}

~Face()

{

if(vert !=NULL)

{

delete[] vert;

vert = NULL;

}

nVerts = 0;

}

};

class Mesh

{

public:

int numVerts;

Point3\* pt;

int numFaces;

Face\* face;

public:

Mesh()

{

numVerts = 0;

pt = NULL;

numFaces = 0;

face = NULL;

}

~Mesh()

{

if (pt != NULL)

{

delete[] pt;

}

if(face != NULL)

{

delete[] face;

}

numVerts = 0;

numFaces = 0;

}

void DrawWireframe();

void DrawColor();

void CreateTetrahedron();

void CreateCube(float fSize);

};

Hàm DrawWireFrame() thực hiện công việc vẽ khung dây của lưới đa giác. Hàm DrawColor() thực hiện công việc vẽ tô màu lưới đa giác.

void Mesh::DrawWireframe()

{

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

for (int f = 0; f < numFaces; f++)

{

glBegin(GL\_POLYGON);

for (int v = 0; v < face[f].nVerts; v++)

{

int iv = face[f].vert[v].vertIndex;

glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);

}

glEnd();

}

}

void Mesh::DrawColor()

{

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

for (int f = 0; f < numFaces; f++)

{

glBegin(GL\_POLYGON);

for (int v = 0; v < face[f].nVerts; v++)

{

int iv = face[f].vert[v].vertIndex;

int ic = face[f].vert[v].colorIndex;

glColor3f(ColorArr[ic][0], ColorArr[ic][1], ColorArr[ic][2]);

glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);

}

glEnd();

}

}

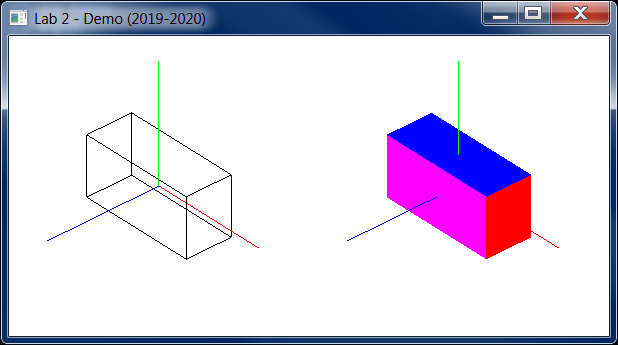
Các hàm CreateTetrahedron(), CreateCube()tạo lưới đa giác biểu diễn các đối tượng hình học hình tứ diện, hình lập phương.

**PHẦN 2 : Thiết kế một số đối tượng hình học cơ bản**

Trong phần này, sinh viên thêm vào lớp Mesh các hàm tạo một số khối hình học.

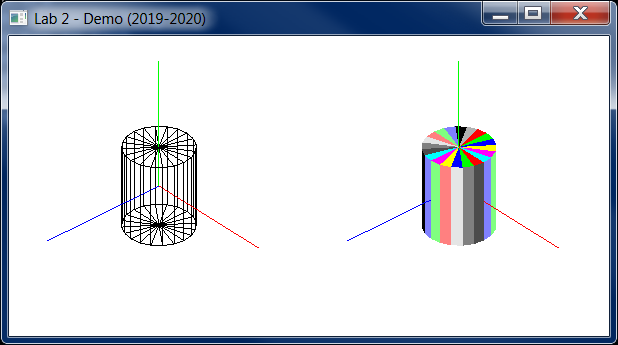
1. **HÌNH HỘP CHỮ NHẬT**

Hãy thêm vào lớp Mesh hàm void Mesh::CreateCuboid(float fSizeX, float fSizeY, float fSizeZ) để tạo lưới đa giác mô phỏng hình hộp chữ nhật. Trong đó fSizeX, fSizeY và fSizeZ lần lượt là kích thước của hình hộp đo theo 3 trục X, Y và Z.



1. **HÌNH TRỤ**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng hình trụ.



Sinh viên có thể tham khảo đoạn mã sau:

void Mesh::CreateCylinder(int nSegment, float fHeight, float fRadius){

numVerts=nSegment\*2 + 2;

pt = new Point3[numVerts];

int i;

int idx;

float fAngle = 2\*PI/nSegment;

float x, y, z;

pt[0].set(0, fHeight/2, 0);

for(i = 0; i<nSegment; i++){

x = fRadius\* cos(fAngle\*i);

z = fRadius\* sin(fAngle\*i);

y = fHeight/2;

pt[i+1].set(x, y, z);

y = -fHeight/2;

pt[i +1 + nSegment].set(x, y, z);

}

pt[numVerts-1].set(0, -fHeight/2, 0);

numFaces= nSegment\*3;

face = new Face[numFaces];

idx = 0;

for(i = 0; i<nSegment; i++){

face[idx].nVerts = 3;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[0].vertIndex = 0;

if(i < nSegment -1)

face[idx].vert[1].vertIndex = i + 2;

else

face[idx].vert[1].vertIndex = 1;

face[idx].vert[2].vertIndex = i + 1;

idx++;

}

for(i = 0; i<nSegment; i++){

face[idx].nVerts = 4;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[0].vertIndex = i+1;

if(i <nSegment - 1)

face[idx].vert[1].vertIndex = i+2;

else

face[idx].vert[1].vertIndex = 1;

face[idx].vert[2].vertIndex = face[idx].vert[1].vertIndex + nSegment;

face[idx].vert[3].vertIndex = face[idx].vert[0].vertIndex + nSegment;

idx++;

}

for(i = 0; i<nSegment; i++){

face[idx].nVerts = 3;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[0].vertIndex = numVerts - 1;

if(i < nSegment -1)

face[idx].vert[2].vertIndex = i + 2 + nSegment;

else

face[idx].vert[2].vertIndex = 1 + nSegment;

face[idx].vert[1].vertIndex = i + 1 + nSegment;

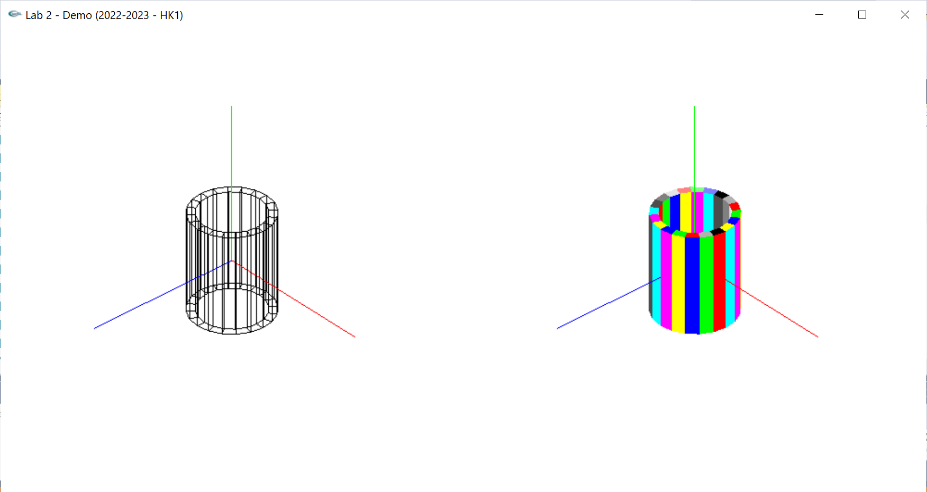
idx++;

}

}

1. **HÌNH TRỤ RỖNG RUỘT**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng hình trụ rỗng ruột.



Sinh viên có thể tham khảo đoạn mã sau.

void Mesh::CreateCylinderWithHole(int nSegment, float fHeight, float fORadius, float fIRadius)

{

numVerts=nSegment\*4;

pt = new Point3[numVerts];

int i;

int idx;

float fAngle = 2\*PI/nSegment;

float x, y, z;

for(i = 0; i<nSegment; i++)

{

x = fORadius\* cos(fAngle\*i);

z = fORadius\* sin(fAngle\*i);

y = fHeight/2;

pt[i].set(x, y, z);

y = -fHeight/2;

pt[i + nSegment].set(x, y, z);

}

for(i = 0; i<nSegment; i++)

{

x = fIRadius\* cos(fAngle\*i);

z = fIRadius\* sin(fAngle\*i);

y = fHeight/2;

pt[i + 2\*nSegment].set(x, y, z);

y = -fHeight/2;

pt[i + 3\*nSegment].set(x, y, z);

}

numFaces= nSegment\*4;

face = new Face[numFaces];

idx = 0;

/////////////// Outside /////////////////////////

for(i = 0; i<nSegment; i++)

{

face[idx].nVerts = 4;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[0].vertIndex = i;

if(i <nSegment - 1)

face[idx].vert[1].vertIndex = i + 1;

else

face[idx].vert[1].vertIndex = 0;

face[idx].vert[2].vertIndex = face[idx].vert[1].vertIndex + nSegment;

face[idx].vert[3].vertIndex = face[idx].vert[0].vertIndex + nSegment;

idx++;

}

/////////////// Inside ///////////////////////

for(i = 2\*nSegment; i<3\*nSegment; i++)

{

face[idx].nVerts = 4;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[1].vertIndex = i;

if(i <3\*nSegment - 1)

face[idx].vert[0].vertIndex = i + 1;

else

face[idx].vert[0].vertIndex = 2\*nSegment;

face[idx].vert[2].vertIndex = face[idx].vert[1].vertIndex + nSegment;

face[idx].vert[3].vertIndex = face[idx].vert[0].vertIndex + nSegment;

idx++;

}

//////////////// Top ////////////////////////////

for(i = 0; i<nSegment; i++)

{

face[idx].nVerts = 4;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[1].vertIndex = i;

if(i <nSegment - 1)

face[idx].vert[0].vertIndex = i + 1;

else

face[idx].vert[0].vertIndex = 0;

face[idx].vert[2].vertIndex = face[idx].vert[1].vertIndex + 2\*nSegment;

face[idx].vert[3].vertIndex = face[idx].vert[0].vertIndex + 2\*nSegment;

idx++;

}

//////////////// Bottom ////////////////////////

for(i = 0; i<nSegment; i++)

{

face[idx].nVerts = 4;

face[idx].vert = new VertexID[face[idx].nVerts];

face[idx].vert[0].vertIndex = i + nSegment;

if(i <nSegment - 1)

face[idx].vert[1].vertIndex = i + nSegment + 1;

else

face[idx].vert[1].vertIndex = nSegment;

face[idx].vert[2].vertIndex = face[idx].vert[1].vertIndex + 2\*nSegment;

face[idx].vert[3].vertIndex = face[idx].vert[0].vertIndex + 2\*nSegment;

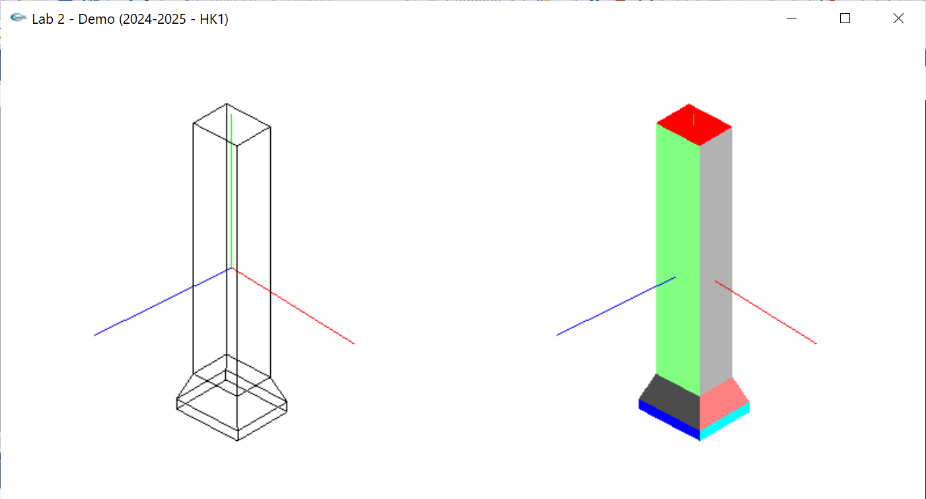
idx++;

}

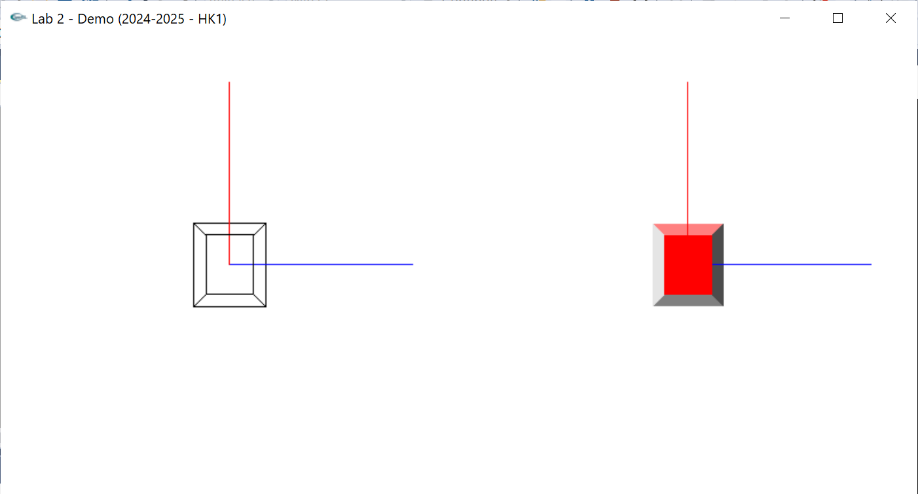
}

1. **SHAPE 1**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng Shape1. Tự xác định tham số của hàm, để có thể tùy biến các kích thước của hình.

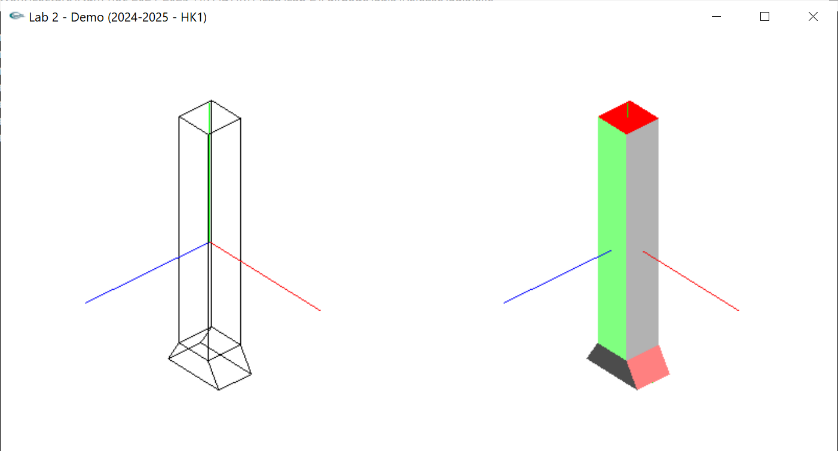


Nhìn từ trên xuống Shape 1 như sau:

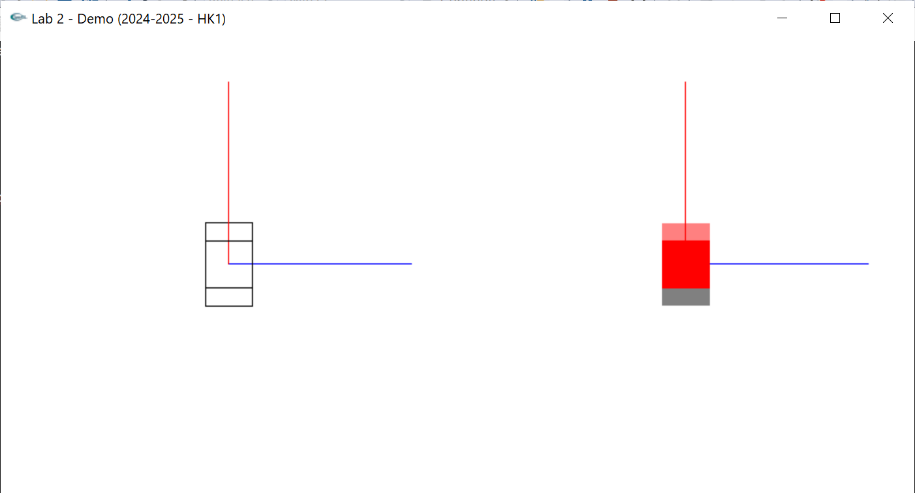


1. **SHAPE 2**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng Shape2. Tự xác định tham số của hàm, để có thể tùy biến các kích thước của hình.

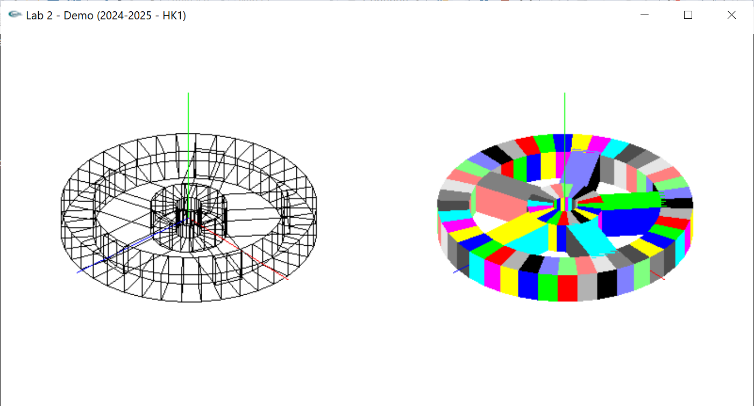


Nhìn từ trên xuống Shape 2 như sau:

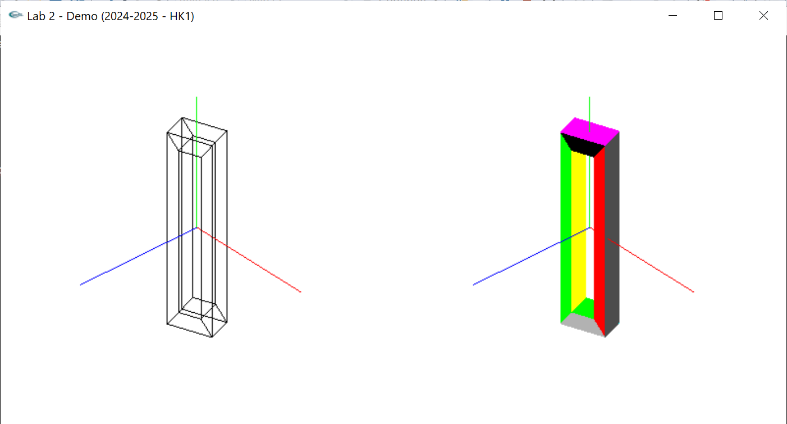


1. **SHAPE 3**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng Shape3. Tự xác định tham số của hàm, để có thể tùy biến các kích thước của hình.



1. **SHAPE 4**

Hãy thêm vào lớp Mesh hàm tạo lưới đa giác mô phỏng Shape4. Đây là khối hình hộp chữ nhật có một khe ở bên trong. Khe này cũng là hình hộp chữ nhật. Tự xác định tham số của hàm, để có thể tùy biến các kích thước của hình.

Hình Shape 1, Shape 2, Shape 3 và Shape 4 (cùng với một số khối hình học khác) được sử dụng để lắp ghép nên mô hình cơ khí sau:

