**Min값?**

**선택한 오브젝트에 딱 맞는 박스를 만들고** **박스의 꼭지점에서 가장 작은 값을 가지는 위치를 Min값이라고 볼 수 있다.**

텍스트, 스크린샷, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Max값?**

**선택한 오브젝트에 딱 맞는 박스를 만들고** **박스의 꼭지점에서 가장 큰 값을 가지는 위치를 Max값이라고 볼 수 있다.**

텍스트, 스크린샷, 그래픽 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**Normal**

**점/선/면이 바라보는 방향**

**getNormal**

**빠르게 지정한 Vertex의 Normal값을 찾을 수 있는 명령어**

(getNormal명령어는 Mesh에서만 사용 할 수 있는 명령어라 현재 poly구조를 잠깐 Mesh로 변경을 해주어야 한다)

Addmodifier명령어를 사용해서 모디파이어에 Edit\_Mesh를 추가

예를 들어

**addMod = Edit\_Mesh ()**

**Addmodifier $ addMod**

작성하면 선택한 오브젝트의 Edit Mesh를 넣어서 poly를 Mesh로 변경가능

**deletemodifier $ 1** 이라고 작성하면

모디파이어에서 제일 위에 있는 것을 삭제가능

**GetVertSelection**

**지정한 오브젝트에서 선택한 vertex의 ID를 bitarray로 알 수 있는 명령어**

Graphical user interface

Description automatically generated

**matrixFromNormal**

**Normal값을 Matrix값으로 변환**

**Matrix?**

Matrix는 쉽게 이야기 해서 3차원 공간에서 회전 값과 위치값을 표현하는 방식

**Matrix구조?**

Matrix값의 초록 상자 안 앞의 3개의 vector값이 회전 값을 표시

뒤에 보라색 상자 안 마지막 Vector값이 위치 값을 표시

Graphical user interface

Description automatically generated

**Getnodebyname**

**지정한 문자열를 이름으로 가지고있는 노드가 있다면 노드 관련 정보 값들을 알수있는 명령어**

Graphical user interface, diagram

Description automatically generated

**BoneSys.createBone**

**bone을 만들수있는 명령어**

Graphical user interface

Description automatically generated

**버튼 기능**

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

**Pivot-center :** 선택한 오브젝트의 pivot을 오브젝트 중심으로 이동

**Pivot-XYCenter and Z\_Zero :** 선택한 오브젝트의 pivot의 XY위치는 오브젝트 중심으로 Z축은 0으로 이동

**Pivot-XYZero :** 선택한 오브젝트의 pivot의 XYZ의 위치를 모두 0으로 이동  
**Min\_Max Pivot :** 기존의 Pivot위치에 설정 한 축의 Min값의 위치나 Max값의 위치로 축을 이동 시켜주는 기능

**Point01 / Bone01체크버튼은 한개만 활성화 될수있고 동시에 두개가 활성 불가능**

**Point / Bone 체크박스를 선택을 하면 아래 있는 Pivot Move버튼을 클릭했을때**

**설정한 오브젝트( Point / Bone )가 생성**

**object체크버튼 :** 지정한 오브젝트의 이름을 등록 할 수 있으며 등록한 오브젝트가 Pivot Move버튼을 클릭 했을때 생성

**Point / Line / Face 체크버튼이 동시에 여러개가 활성화 되지 않고 3개의 체크박스중에 무조건 한개는 활성화**

**Point / Line / Face :** 원하는 부분을 선택하고 pivot move버튼을 클릭하면 선택한 점/선/면 중에 하나로 pivot이 이동

**N 체크박스도 활성화 되어있는 상태에 Pivot Move버튼을 클릭하면 새로 생성된 오브젝트가 선택한 점선면의 normal방향을 인식해서 생성**

try(destroydialog PTC)catch()

rollout PTC "Pivot Controller"

(

group "Object\_type"

(

button OB01 "Pivot-Center" align:#left across:3

button OB02 "Pivot-XYCenter and Z\_Zero" align:#center

button OB03 "Pivot-XYZero" align:#right

)

group "Min\_Max Pivot"

(

button PXmax01 "Plus X\_max" width:150 across:2

button PXmin01 "Plus X\_min" width:150

button PYmax01 "Plus Y\_max" width:150 across:2

button PYmin01 "Plus Y\_min" width:150

button PZmax01 "Plus Z\_max" width:150 across:2

button PZmin01 "Plus Z\_min" width:150

)

Group "Make object"

(

checkbox Point01 "Point" across:3

checkbox Bone01 "Bone"

checkButton selob01 "object" width:90

)

group "Vertex and Edge and Face\_Type"

(

checkbox Po01 "Point" across:5

checkbox Li01 "Line"

checkbox Fa01 "Face"

checkbox Normal01 "N"

button PMC01 "Pivot move"

)

fn Pivotmove axis =

(

local sellist = selection as array

if sellist.count > 0 do

(

if axis == 1 do (for i in sellist do i.pivot = i.center)

if axis == 2 do (for i in sellist do i.pivot = [i.center.x,i.center.y,0])

if axis == 3 do (for i in sellist do i.pivot = [0,0,0])

)

)

fn Pivotminmax axis =

(

local sellist = selection as array

if sellist.count > 0 then

(

if axis == 1 do (for i in sellist do i.pivot.x = i.max.x)

if axis == 2 do (for i in sellist do i.pivot.x = i.min.x)

if axis == 3 do (for i in sellist do i.pivot.y = i.max.y)

if axis == 4 do (for i in sellist do i.pivot.y = i.min.y)

if axis == 5 do (for i in sellist do i.pivot.z = i.max.z)

if axis == 6 do (for i in sellist do i.pivot.z = i.min.z)

)

else( messagebox "check select object")

)

fn wantpivot obj =

(

subobjectLevel = 1

sels = getVertSelection obj

Asels = sels as array

if classof obj == Editable\_Poly do

(

addV = [0,0,0]

for v in 1 to Asels.count do

(

addV += polyop.getVert obj Asels[v]

)

global pos = addV/Asels.count

if Point01.state == true do

(

TESTBox = Point()

TESTBox.Box = on

TESTBox.name = uniquename "HSB\_Point"

TESTBox.pos = pos

)

if Bone01.state == true do

(

TESTBox = BoneSys.createBone [0,0,0] [5,0,0] [0,0,1]

TESTBox.width = 5

TESTBox.height = 5

TESTBox.boneEnable=false

TESTBox.name = uniquename "HSB\_Bone"

TESTBox.pos = pos

)

if selob01.state == true and selob01.text != "object" do

(

if getnodebyname selob01.text != undefined do

(

TESTBox = getnodebyname selob01.text

TESTBox.pos = pos

)

)

if Normal01.state == true do

(

vertArray = polyOp.getVertSelection obj as array

tNormal = [0,0,0]

addMod = Edit\_Mesh ()

addmodifier obj addMod

for v in vertArray do(tNormal += getNormal obj v)

deletemodifier obj 1

Global SelNormal = matrixFromNormal(tNormal / vertArray.count)

TESTBox.transform = SelNormal

TESTBox.pos = pos

)

max create mode

)

)

on PTC open do

(

Po01.state = true

)

on Point01 changed state do

(

if Point01.state == true do

(

Bone01.state = false

selob01.state = false

selob01.text = "object"

)

)

on Bone01 changed state do

(

if Bone01.state == true do

(

Point01.state = false

selob01.state = false

selob01.text = "object"

)

)

on selob01 changed state do

(

if selob01.state == true then

(

Point01.state = false

Bone01.state = false

if (selection as array).count == 1 do

(

selob01.text = $.name

)

)

else

(

selob01.text = "object"

)

)

on Po01 changed state do

(

if Po01.state == true do

(

Li01.state = false

Fa01.state = false

)

if Po01.state == false do

(

if Li01.state == false and Fa01.state == false do

(

Po01.state = true

)

)

)

on Li01 changed state do

(

if Li01.state == true do

(

Po01.state = false

Fa01.state = false

)

if Li01.state == false do

(

if Po01.state == false and Fa01.state == false do

(

Li01.state = true

)

)

)

on Fa01 changed state do

(

if Fa01.state == true do

(

Po01.state = false

Li01.state = false

)

if Fa01.state == false do

(

if Po01.state == false and Li01.state == false do

(

Fa01.state = true

)

)

)

on OB01 pressed do

(

Pivotmove 1

)

on OB02 pressed do

(

Pivotmove 2

)

on OB03 pressed do

(

Pivotmove 3

)

on PMC01 pressed do

(

try

(

sellist = selection as array

if sellist.count ==1 do

(

obj = $

if Po01.state == true do

(

Vertsels = getVertSelection obj

VertAsels = Vertsels as array

if VertAsels.count > 0 do

(

wantpivot obj

)

)

if Li01.state == true do

(

Edgesels = getEdgeSelection obj

EdgeAsels = Edgesels as array

if EdgeAsels.count >0 do

(

obj.EditablePoly.ConvertSelection #Edge #Vertex

wantpivot obj

)

)

if Fa01.state == true do

(

Facesels = getFaceSelection obj

FaceAsels = Facesels as array

if FaceAsels.count > 0 do

(

obj.EditablePoly.ConvertSelection #Face #Vertex

wantpivot obj

)

)

obj.pivot = pos

)

)

catch()

)

on PZmax01 pressed do

(

Pivotminmax 5

)

on PZmin01 pressed do

(

Pivotminmax 6

)

on PXmax01 pressed do

(

Pivotminmax 1

)

on PXmin01 pressed do

(

Pivotminmax 2

)

on PYmax01 pressed do

(

Pivotminmax 3

)

on PYmin01 pressed do

(

Pivotminmax 4

)

)

createdialog PTC width:340