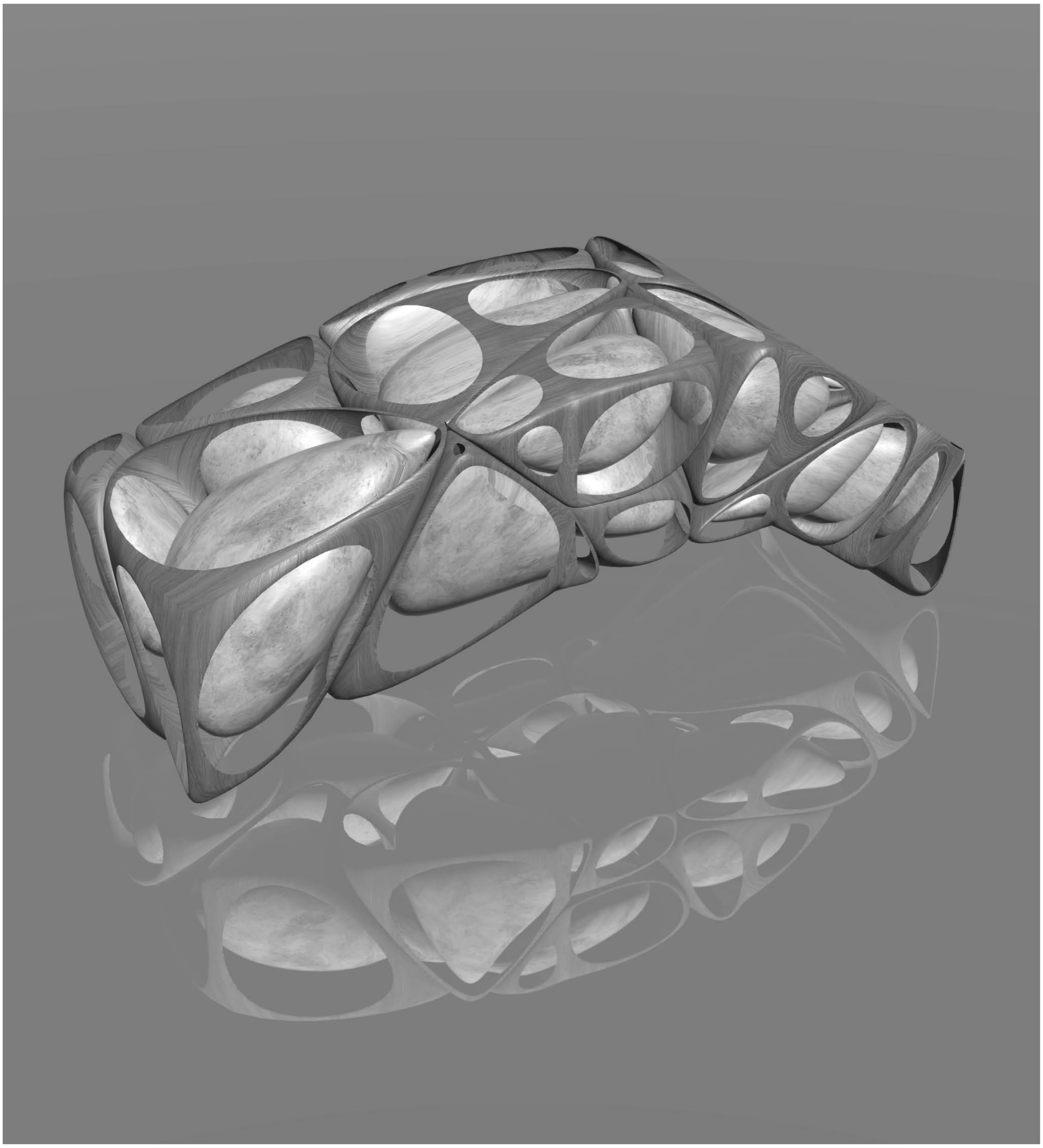
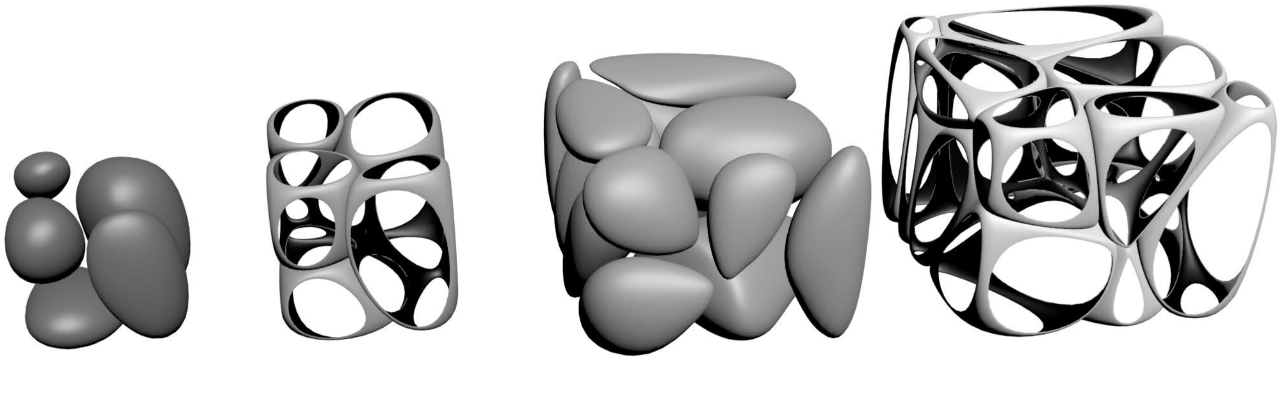
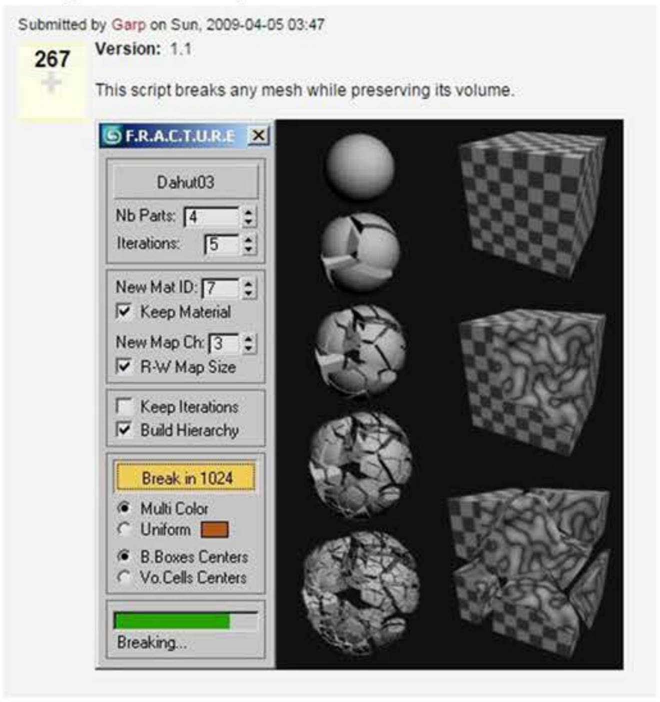
DPM3 Project 1 -
Voronoi Bone Structure and Pebble Generator





Original Script - Fracture Voronoi


Submitted by Garp on Sun, 2009-04-05 03:47

267

Version: 1.1

This script breaks any mesh while preserving its volume.

F.R.A.C.T.U.R.E

Dahut03

Nb Parts: 4

Iterations: 5

New Mat ID: 7

☑ Keep Material

New Map Ch: 3

☑ R-W Map Size

☐ Keep Iterations

☑ Build Hierarchy

Break in 1024

● Multi Color

○ Uniform

● B.Boxes Centers

○ Vo.Cells Centers

Breaking...

This is the original script that we extracted from which was found on
http://www.scriptspot.com/3ds-max/scripts/fracture-voronoi

This script divides any object into individual Voronoi Cells and the user is able to specify the number of fragments and iterations.
 - original UVs are preserved and 'projected' onto the new faces;
- simple planar mapping is applied to a new channel and consistent throughout the new parts;
- new material ID is applied to the new faces;
- can keep intermediate generations;
- can build hierarchy.

```
(   -- start script

global rltFractureVoronoi
try destroyDialog rltFractureVoronoi catch()

rollout rltFractureVoronoi "F.R.A.C.T.U.R.E"
(
    groupBox boxSetUp "" pos:[5,0] width:116 height:83
    fn geometryFilter obj = superClassOf obj == GeometryClass
    pickButton pbObject "Pick Object" pos:[10,11] width:106 height:25 filter:geometryFilter tooltip:"object to fracture"
    spinner spnNbParts "Nb Parts: " pos:[09,41] width:106 height:16 range:[2,1000,10] type:#integer enabled:false
    spinner spnNbIter "Iterations:     " pos:[28,61] width:87 height:16 range:[1,10,1] type:#integer enabled:false

    groupBox boxMat "" pos:[5,81] width:116 height:89
    spinner spnNewID "New Mat ID:" pos:[24,93] width:91 height:16 range:[1,100,1] indeterminate:true type:#integer enabled:false
    checkBox cbKeepMat "Keep Material" pos:[12,110] checked:true enabled:false
    spinner spnNewCh "New Map Ch:" pos:[32,133] width:83 height:16 range:[1,10,3] type:#integer enabled:false
    checkBox cbRWMS "R-W Map Size" pos:[12,150] checked:true enabled:false

    groupBox boxHierarchy "" pos:[5,168] width:116 height:48
    checkBox cbKeepGen "Keep Iterations" pos:[12,179] checked:false enabled:false
    checkBox cbLinkGen "Build Hierarchy" pos:[12,196] checked:false enabled:false

    groupBox boxCreate "" pos:[5,214] width:116 height:109
    button btnCreate "Break in 10" pos:[10,225] width:106 height:25 tooltip:"pick object first" enabled:false
    radiobuttons rdoColor "" pos:[12,253] width:72 height:32 labels:#("Multi Color","Uniform") default:1 columns:1 enabled:false
    colorPicker cpParts "" pos:[70,270] fieldWidth:20 height:12 visible:false
    radiobuttons rdoCenter "" pos:[12,288] width:72 height:32 labels:#("B.Boxes Centers","Vo.Cells Centers") default:1 columns:1 enabled:false

    groupBox boxProgress "" pos:[5,321] width:116 height:49
    progressBar pbProgress "" pos:[10,335] width:106 height:15 value:0 color:[0,96,0]
    label lblProStatus "" pos:[10,351] width:100 height:17

    local theObject         -- holds the original object


    on pbObject picked obj do
    (
        pbObject.text = obj.name
        theObject = obj
        spnNbParts.enabled = true
        spnNbIter.enabled = true
        spnNewID.enabled = true
        cbKeepMat.enabled = true
        spnNewCh.enabled = true
        cbRWMS.enabled = true
        cbLinkGen.enabled = true
        btnCreate.enabled = true
        btnCreate.tooltip = "start creating parts"
        rdoColor.enabled = true
        rdoCenter.enabled = true
        cpParts.color = obj.wireColor
        cpParts.visible = true

        when obj deleted do
        (
            btnCreate.enabled = false
            btnCreate.tooltip = pbObject.text + " has been deleted!"
            pbObject.text = "Pick Object"
        )
        undo off
        (   -- gets new mat ID for new faces
            m = edit_mesh()
            addModifier obj m
            spnNewID.value = amax(for i = 1 to obj.numfaces collect getFaceMatID obj i) + 1
            deleteModifier obj m
        )

    )   -- end on btnMesh picked theMesh


    on btnCreate pressed do
    (
        undo off
        (
            disableSceneRedraw()
            clearSelection()
            start = timeStamp()

            local nbParts = spnNbParts.value
            local nbIter = spnNbIter.value
            local keepGen = cbKeepGen.checked
            local linkGen = cbLinkGen.checked
            local aPartsStart = #()
            local aPartsEnd = #()
            local aAllParts = #()
            local aAllCoords = #()
            local thePlane = plane width:1 length:1 widthSegs:1 lengthSegs:1    -- plane helper for slice plane
            local theMesh = editable_mesh()
            local abortBreaking = false

            lblProStatus.caption = " Breaking..."

            --  BREAKING UP
            ---------------

            -- clean copy (no custom attributes, keyframes, weird transforms, etc
            theCopy = copy theObject
            theCopy.name = "toto"
            resetXForm theCopy
            convertToMesh theCopy
            theMesh.mesh = theCopy.mesh
            theMesh.transform = theCopy.transform
            theMesh.pivot = [0,0,0]
            resetXForm theMesh
            convertToMesh theMesh
            delete theCopy
```

Creates a copy of the orihinal object which the script works off

```
            -- material and UVs
            if cbKeepMat.checked do theMesh.material = theObject.material
            addModifier theMesh (uvwMap mapChannel:spnNewCh.value realWorldMapSize:cbRWMS.checked)
            convertToMesh theMesh
            setFaceSelection theMesh #{}

            -- parts creation
            aPartsEnd = #(theMesh)
            for iter = 1 to nbIter while not abortBreaking do
            (
                aPartsStart = aPartsEnd
                aPartsEnd = #()

                for obj in aPartsStart while not abortBreaking do
                (
                    aPartsTemp = for i = 1 to nbParts collect copy obj
                    pSys = pcloud emitter:obj formation:3 total_number:nbParts quantityMethod:1 viewPercent:100 seed:(random 0 100)
                    aCoords = for i = 1 to nbParts collect particlePos pSys i-- fill with random coordinates
                    delete pSys
                    for i = 1 to nbParts - 1 do for j = i + 1 to nbParts while not abortBreaking do-- for each pair of coords
                    (
                        thePlane.pos = (aCoords[i] + aCoords[j]) / 2
                        thePlane.dir = aCoords[j] - aCoords[i]

                        addModifier aPartsTemp[i] (sliceModifier slice_type:2)
                        addModifier aPartsTemp[j] (sliceModifier slice_type:3)
                        aPartsTemp[i].slice.slice_plane.transform = thePlane.transform
                        aPartsTemp[j].slice.slice_plane.transform = thePlane.transform
                        addModifier aPartsTemp[i] (cap_holes())
                        addModifier aPartsTemp[j] (cap_holes())
                        convertToMesh aPartsTemp[i]
                        convertToMesh aPartsTemp[j]

                        if keyboard.escPressed do abortBreaking = queryBox "Do you want to abort and delete already created parts?"
                    )   -- end i loop
                    aPartsEnd += aPartsTemp
                    aAllParts += aPartsTemp
                    aAllCoords += aCoords

                    total = nbParts * ((nbParts^nbIter - 1) / (nbParts - 1))
                    prog = 100 * aAllParts.count / total
                    pbProgress.value = prog
                    pbProgress.color = [200 - prog * 2,prog * 2,0]
                )   -- end obj loop
            )   -- end iter loop
```

Creation of parts using voroni tessellation at the end of the process object parts are put into an array called aAllParts

```
            if not abortBreaking then
            (
                lblProStatus.caption = " Finalizing..."

                --  TIDYING UP
                --------------

                delete theMesh
                delete thePlane
                hide theObject
```

Deletes the helper plane and mesh which were used for the parts creation

```
                -- intermediate generations
                if not keepGen and nbIter != 1 do
                (
                    ind = 0
                    for i = 1 to nbIter - 1 do for j = 1 to nbParts^i do
```

```
        (
            ind += 1
            delete aAllParts[ind]
            aAllCoords[ind] = undefined
            )
            aAllParts = for obj in aAllParts where not isDeleted obj collect obj
            aAllCoords = for c in aAllCoords where c != undefined collect c
        )

        -- coordinates
        if rdoCenter.state == 1 then centerPivot aAllParts
        else for i = 1 to aAllParts.count do aAllParts[i].pivot = aAllCoords[i]
        resetXForm aAllParts
        convertToMesh aAllParts

        -- new faces ID
        newID = spnNewID.value
        for obj in aAllParts do
        (
            for f in getFaceSelection obj do setFaceMatID obj f newID
            setFaceSelection obj #{}
        )

        -- names
        if not keepGen or nbIter == 1 then
            for i = 1 to aAllParts.count do aAllParts[i].name = theObject.name + "_Part_" + i as string
        else
        (
            for i = 1 to nbParts do aAllParts[i].name = theObject.name + "_Part_" + i as string
            indP = 0
            indC = nbParts
            for i = 1 to nbIter - 1 do for j = 1 to nbParts^i do
            (
                indP += 1
                for k = 1 to nbParts do
                (
                    indC += 1
                    aAllParts[indC].name = aAllParts[indP].name + "_" + k as string
                )   -- end k loop
            )   -- end j loop
        )   -- end else

        -- layers
        -- (comment out this block if you don't want any layer, intermediate generations will not be hidden)
        -- (FROM HERE...)
        if not keepGen or nbIter == 1 then
        (
            if layerManager.getLayerFromName (theObject.name + "_Parts") == undefined then
                theLayer = layerManager.newLayerFromName (theObject.name + "_Parts")
            else theLayer = layerManager.getLayerFromName (theObject.name + "_Parts")
            for obj in aAllParts do theLayer.addNode obj
        )   -- end if
        else
        (
            aTheLayers = for i = 1 to nbIter collect
            (
                if layerManager.getLayerFromName (theObject.name + "_Gen_" + i as string) == undefined then
                    layerManager.newLayerFromName (theObject.name + "_Gen_" + i as string)
                else layerManager.getLayerFromName (theObject.name + "_Gen_" + i as string)
            )
            for i = 1 to nbIter - 1 do aTheLayers[i].isHidden = true
            ind = 0
            for i = 1 to nbIter do for j = 1 to nbParts^i do
            (
                ind += 1
                aTheLayers[i].addNode aAllParts[ind]
            )   -- end i loop
        )   -- end else
        -- (...TO HERE)

        -- hierarchy
        if linkGen do
        (
            if not KeepGen or nbIter == 1 then for obj in aAllParts do attachObjects theObject obj move:false
            else
            (
                for i = 1 to nbParts do attachObjects theObject aAllParts[i] move:false
                indP = 0
                indC = nbParts
                for i = 1 to nbIter - 1 do for j = 1 to nbParts^i do
                (
                    indP += 1
                    for k = 1 to nbParts do
                    (
                        indC += 1
                        attachObjects aAllParts[indP] aAllParts[indC] move:false
                    )   -- end k loop
                )   -- end j loop
            )   -- end else
        )   -- end if linkGen

        -- colors
        if rdoColor.state == 1 then for obj in aAllParts do obj.wireColor = random black white
        else aAllParts.wireColor = cpParts.color

        lblProStatus.caption = " Done in " + (formattedPrint ((timeStamp() - start) / 1000.0) format:".1f") + "sec."

        enableSceneRedraw()
        completeRedraw()
    )
    else
    (
        delete thePlane
        delete theMesh
        delete aAllParts
        pbProgress.value = 0
        lblProStatus.caption = " Stopped"
        enableSceneRedraw()
    )   -- end test abortBreaking

)   -- end undo off
```

```
    )  -- end btnCreate pressed


    on spnNbParts changed val do
    (
        btnCreate.caption = "Break in " + ((val ^ spnNbIter.value) as string)
    )


    on spnNbIter changed val do
    (
        btnCreate.caption = "Break in " + ((spnNbParts.value ^ val) as string)
        cbKeepGen.enabled = val != 1
    )


    on rltFractureVoronoi close do
    (
        enableSceneRedraw()
        CompleteRedraw()
        callbacks.removeScripts id:#FVcbID01
    )

)  -- end rollout rltFractureVor


createDialog rltFractureVoronoi 126 375 60 130

)  -- end script
```

The areas highlighted in yellow are the parts that we extracted from the Fracture Voronoi script. The extracted script allows us to 'break' an object into individual parts according to the Voronoi tessellation method resulting in Voronoi Cells

Writing the Script

We were originally interested in using a tessellation pattern to subdivide and tile a chosen surface. The idea was that the subdivisions or individual tiles could be extruded and controlled with a control point. However we soon decided it would be more interesting to try and script something that would instead of tiles create more of a frame structure using a tessellation pattern on a chosen surface.We started by looking into tessellation patterns but found that they were too regular with not much variation. Therefore we decided to go with a voronoi tessellation pattern as it was a pattern that seemingly looks like random subdivisions but it partitions itself into voronoi cells according to the distance 'closeness' to set points in a plane.

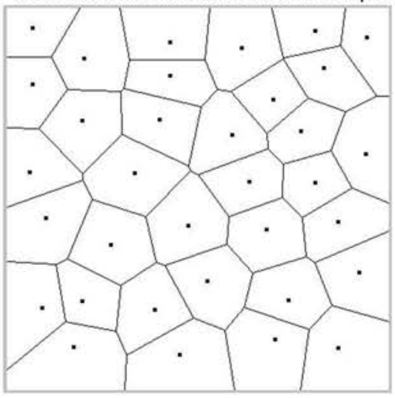


Image Source: http://www.cs.wustl.edu/~pless/546/lectures/L11.html

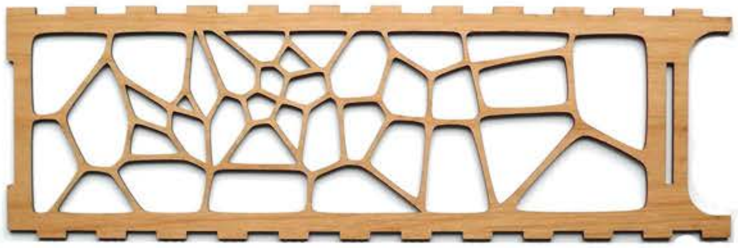


Image Source: http://cargocollective.com/kimcovey/Voronoi-Structure-Lamp

The maths behind the voronoi fragments was rather difficult but we managed to find an existing script (fracture voronoi) that we could extract from which we could use to create voronoi cells. This lead on to looking into how we would go about writing the script.

Our initial idea was as follows:
1. To create a plane/object which we could give a thickness
2. Divide the plane/object into voronoi cells using the extracted script and be able to specify the number of fragments
3. Select all the edges and vertices and weld/attach them together to create one singular frame
4. Delete inner faces to just be left with a frame
5. Give the frame a thickness using shell modifier or something similar

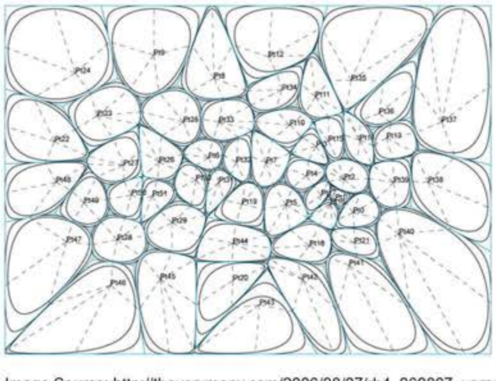


Image Source: http://theverymany.com/2006/08/07/rh4_060807_vornoi/

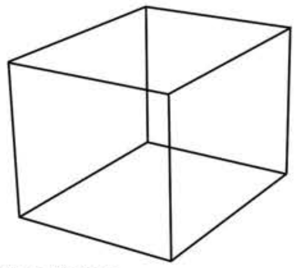
Then we found this image online and thought that the structure could be given a smoothness of some which would make the end result much cleaner and pleasent to look at.So having set the initial ideas we started scripting and went through several iterations of the final script to reach the end product.

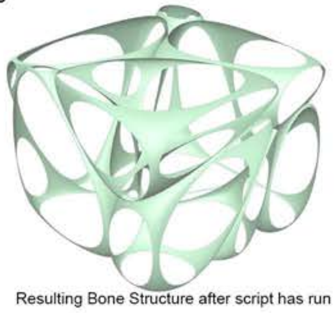
Our final script enables the user to pick any object and choose the number of voronoi cells that it should fragment into. Then the fragments can be used to create a voronoi bone like structure and also voronoi pebbles 'cells' derived from the same fragments

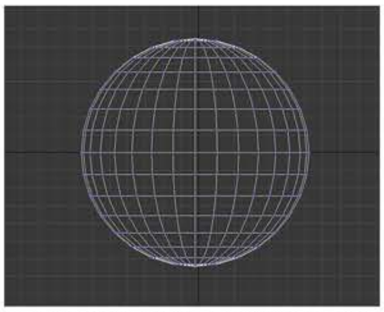
Final Script- Voronpi Bone Structure/Pebble Generator

```
  global theObject= $
  global nbParts = 5


rollout generateBox"Generate Box" --define a rollout
  (

  struct myFunc
  (

     fn removeRandomEdges =
        (
              theObject = $
              theObject = convertToPoly (theObject)
              local amtOfEdges
              amtOfEdges = polyOp.getNumEdges theObject               --the number of edges in the poly. Includes any dead edges.
              local edgesToDelete = amtOfEdges/5                   --1/5 of the edges to be removed
              local randomEdge

              for i = 1 to edgesToDelete do                     --loop picking random edges to delete
                (
                   amtOfEdges = polyOp.getNumEdges theObject
                   select theObject
                   max modify mode
                   subObjectLevel = 2
                   randomEdge = random 1 amtOfEdges
                   $.EditablePoly.SetSelection #Edge #{randomEdge}      --set the selection in the specified sub-object level
                   $.EditablePoly.Remove ()                          --remove edges
                )
        ),
```

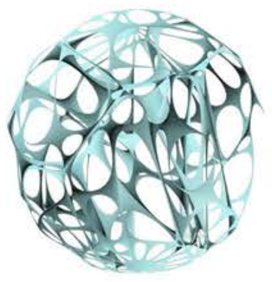
The first function we wrote was the random edges. This part of the script allows us to go through the number of edges in a picked object and randomly remove 1/5 of the edges. The reason that this was necessary was that with objects like a cube have a small number of edges,verticies and faces which allows for a pleasant bone structure. On the other hand shapes like a shere when created in 3DsMax have a very large number of edges and verticies so when the script runs the resultant bone structure is very regular and not random.


Cube-Wireframe


Resulting Bone Structure after script has run


Sphere-Wireframe showing large number of edges, vertices and faces

Bone Structure no Random Edges Removed


Bone Structure with Random Edges Removed

As you can see the sphere structure has many edges and vertices so with no edges removed the resulting structure is very regular compared to that of a cube. With the Random Edges removed the sphere is less regular with more random spacings. So the idea was that the user can choose to remove edges of certain shapes so that when the script runs and generates a bone structure the end result is more pleasing to the eye.

```

        fn fragmentation =
        (
              global theMesh = editable_mesh()
              global thePlane = plane width:1 length:1 widthSegs:1 lengthSegs:1  -- plane helper for slice plane
              global aPartsStart = #()
              global aPartsEnd = #()
              global aAllParts = #()
              global aAllPartsCopy = #()
              global aAllCoords = #()
              global nbIter = 1

           theObject = $
           -- clean copy (no custom attributes, keyframes, weird transforms, etc
              theCopy = copy theObject
              theCopy.name = "desi"
              resetXForm theCopy
```

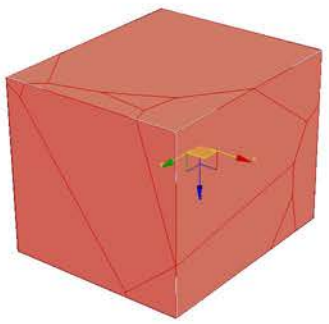
```
            convertToMesh theCopy
            theMesh.mesh = theCopy.mesh
            theMesh.transform = theCopy.transform
            theMesh.pivot = [0,0,0]
            resetXForm theMesh
            convertToMesh theMesh
            delete theCopy

    -- parts creation
            aPartsEnd = #(theMesh)
            for iter = 1 to 1 do
               (
                  aPartsStart = aPartsEnd
                  aPartsEnd = #()

                  for theObject in aPartsStart do
                        (
                           aPartsTemp = for i = 1 to nbParts collect copy theObject
                           pSys = pcloud emitter:theObject formation:3 total_number:nbParts quantityMethod:1 viewPercent:100 seed:(random 0 100)
                           aCoords = for i = 1 to nbParts collect particlePos pSys i -- fill with random coordinates
                           delete pSys
                           for i = 1 to nbParts - 1 do for j = i + 1 to nbParts  do-- for each pair of coords
                                 (
                                    thePlane.pos = (aCoords[i] + aCoords[j]) / 2
                                    thePlane.dir = aCoords[j] - aCoords[i]

                                    addModifier aPartsTemp[i] (sliceModifier slice_type:2)
                                    addModifier aPartsTemp[j] (sliceModifier slice_type:3)
                                    aPartsTemp[i].slice.slice_plane.transform = thePlane.transform
                                    aPartsTemp[j].slice.slice_plane.transform = thePlane.transform
                                    addModifier aPartsTemp[i] (cap_holes())
                                    addModifier aPartsTemp[j] (cap_holes())
                                    convertToMesh aPartsTemp[i]
                                    convertToMesh aPartsTemp[j]

                                 )  -- end i loop
                           aPartsEnd += aPartsTemp
                           aAllParts += aPartsTemp
                           aAllCoords += aCoords

                           total = nbParts * ((nbParts^nbIter - 1) / (nbParts - 1))
                        )  -- end theObject loop
               )  -- end iter loop
            for theObject in aAllParts do theObject.wireColor = random black white
            delete theObject   --st box
            delete thePlane    -- helper plane
            delete theMesh     --thelastcopy

            for obj in aAllParts do convertTo obj Editable_Poly      --conv the parts to edit poly
    ),

```

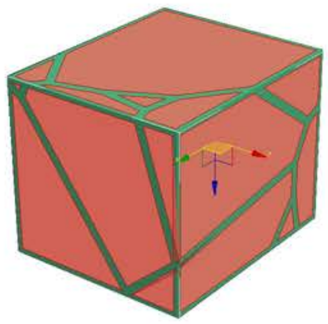
Fragment function allows us to cut a selected object into a specified number of voronoi fragments. This is where all the fragmented parts to be used further in the script are created and then they are stored in an array called aAllParts defined at the beginning of the fragment function. All the parts are then converted into an editable poly so that we can later modify the objects using poly operations.

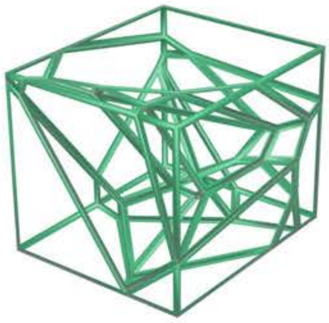
```
    fn generateForm = -- this is the inset and surfsubdivision
       (
              total = aAllParts.count
              for i = 2 to total do
                 (
                    polyop.attach aAllParts[1] aAllParts[i] -- chooses the first object and attaches it to the next
                 )

              aAllPartsCopy = copy aAllParts[1]

              numfaces = polyop.getnumfaces aAllParts[1]

              select aAllParts[1] --select the object
              max modify mode --go to modify panel
              subobjectlevel = 4 --go to polygon sub-object level
              polyop.setfaceselection aAllParts[1] #{1..numfaces}
              aAllParts[1].insetType = 1 --insets by polygon
              aAllParts[1].insetAmount = 1.5  --inset amount
              aAllParts[1].ButtonOp #inset --grow the selection


              polyop.setfaceselection aAllParts[1] #{1..numfaces} --selects all faces
              delete aAllParts[1].selectedfaces --deletes faces
              select aAllParts[1]
              aAllParts[1].surfSubdivide = on --nurms subdivision
              aAllParts[1].iterations = 3

```

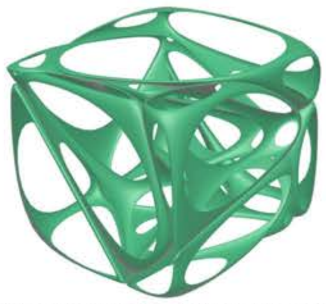
```
                select aAllPartsCopy
                hide aAllPartsCopy  --hide
        ),


```

The generateForm function is what allows us to create the bone structure. First we count the number of objects in the array aAllParts and then do a loop to attach all the objects in the array to each other. The next operation is to select the poly faces and do an inset modifier the incettype =1 which tells maxscript to inset the object by polygons. After the inset operation is done the poly faces are selected again and deleted to leave only the poly frame. To achieve the final smooth bone structure the frame a Nurms Subdivision is applied which smooths the frame. A copy of aAllParts is made within the function and hidden so that it can be used later in the script within another function.


Attached Object with faces selected


Inset by Polygon


Poly faces deleted and resulting frame


Smoothed Frame - Nurms Subdivide

Development of script

Originally the aim of the script was to be able to generate a bone like voronoi structure using any object. But we decided that the script could be taken further and improved. what we decided to do was to pack or fill the voids in the frame with Voronoi 'Pebbles' derived from the same fragments as the original object. This way we would avoid getting random pebbles and have a pebble mass that had a direct relation to the voids.

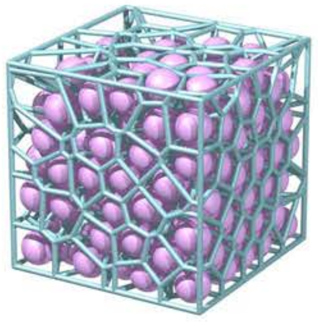
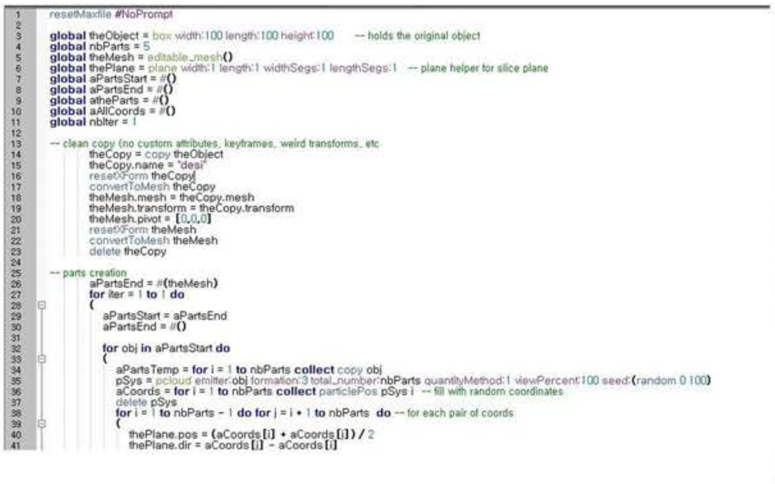
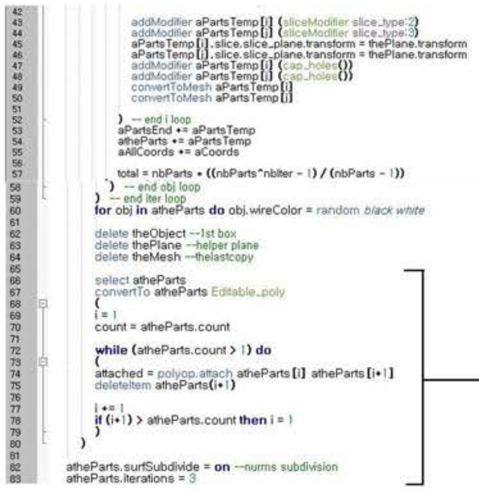


Image Source:http://math.lbl.gov/voro++/examples/radical/
This image was the inspiration behind 'packing' the voids with another object.



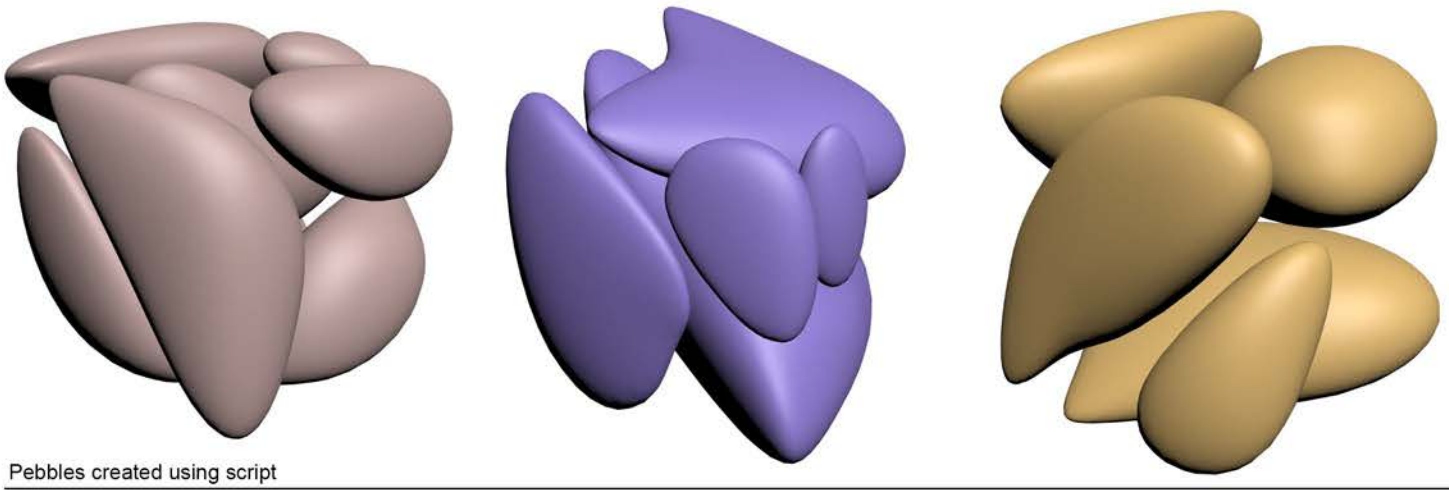
```
resetMaxfile #NoPrompt

global theObject = box width:100 length:100 height:100      -- holds the original object
global nbParts = 5
global theMesh = editable_mesh()
global thePlane = plane width:1 length:1 widthSegs:1 lengthSegs:1   -- plane helper for slice plane
global aPartsStart = #()
global aPartsEnd = #()
global atheParts = #()
global aAllCoords = #()
global nbIter = 1

-- clean copy (no custom attributes, keyframes, weird transforms, etc
        theCopy = copy theObject
        theCopy.name = "desi"
        resetXForm theCopy
        convertToMesh theCopy
        theMesh.mesh = theCopy.mesh
        theMesh.transform = theCopy.transform
        theMesh.pivot = [0,0,0]
        resetXForm theMesh
        convertToMesh theMesh
        delete theCopy

-- parts creation
        aPartsEnd = #(theMesh)
        for iter = 1 to 1 do
        (
          aPartsStart = aPartsEnd
          aPartsEnd = #()

          for obj in aPartsStart do
          (
            aPartsTemp = for i = 1 to nbParts collect copy obj
            pSys = pcloud emitter:obj formation:3 total_number:nbParts quantityMethod:1 viewPercent:100 seed:(random 0 100)
            aCoords = for i = 1 to nbParts collect particlePos pSys i  -- fill with random coordinates
            delete pSys
            for i = 1 to nbParts - 1 do for j = i + 1 to nbParts  do -- for each pair of coords
            (
              thePlane.pos = (aCoords[i] + aCoords[j]) / 2
              thePlane.dir = aCoords[j] - aCoords[i]
```



```
                addModifier aPartsTemp[i] (sliceModifier slice_type:2)
                addModifier aPartsTemp[j] (sliceModifier slice_type:3)
                aPartsTemp[i].slice.slice_plane.transform = thePlane.transform
                aPartsTemp[j].slice.slice_plane.transform = thePlane.transform
                addModifier aPartsTemp[i] (cap_holes())
                addModifier aPartsTemp[j] (cap_holes())
                convertToMesh aPartsTemp[i]
                convertToMesh aPartsTemp[j]

            ) -- end i loop
            aPartsEnd += aPartsTemp
            atheParts += aPartsTemp
            aAllCoords += aCoords

            total = nbParts * ((nbParts^nbIter - 1) / (nbParts - 1))
          ) -- end obj loop
        ) -- end iter loop
        for obj in atheParts do obj.wireColor = random black white

        delete theObject --1st box
        delete thePlane --helper plane
        delete theMesh --thelastcopy

        select atheParts
        convertTo atheParts Editable_poly
        (
        i = 1
        count = atheParts.count

        while (atheParts.count > 1) do
        (
        attached = polyop.attach atheParts[i] atheParts[i+1]
        deleteItem atheParts(i+1)

        i += 1
        if (i+1) > atheParts.count then i = 1
        )
      )

    atheParts.surfSubdivide = on --nurms subdivision
    atheParts.iterations = 3
```

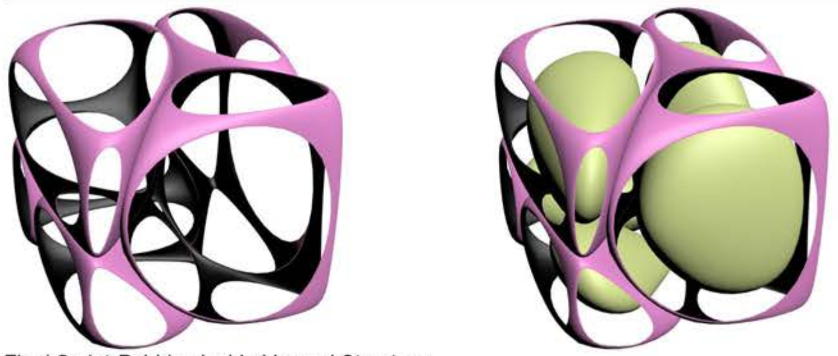
Creates Pebbles

To test the pebble script we created an object and then ran the fragment function to cut the object into parts. Then the fragments are selected and converted into an editable poly and attached. The resulting object is selected and a nurms subdivision is applied to it which creates the pebbles


Pebbles created using script

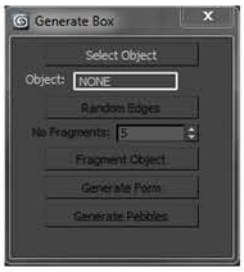
```
fn generatePebbles =
  (
          unhide aAllPartsCopy  --unhide all
          select aAllParts[1]

          aAllPartsCopy.surfSubdivide = on --nurms subdivision
          aAllPartsCopy.iterations = 3
  )
)
```

In the end we realised that we did not need to use such a long script. By doing aAllPartscopy = copy aAllParts and creating a copy of the original fragments stored in the aAllParts array we could call then later and use them to from pebbles which were formed from the same voronoi fragments as the voids. So that we can see the bone structure generation from the original parts the copied parts were hidden after the generateForm function. So when generatePebbles function is called the coped parts are called back and unhidden so that you can view the pebbles as well as the bone structure.

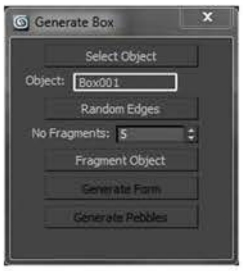

Final Script-Pebbles inside Voronoi Structure

```

        pickButton selectObject "Select Object" width:140 align:#center
        edittext selectObject_tf "Object: " text:"NONE" readonly:true width:138
        button randomEdges_btn "Random Edges" width:140 align:#center enabled:false
        spinner nbParts_spinner "No Fragments: " range:[1,20,nbParts] type:#integer width:140 align:#center enabled:false
        button fragmentation_btn "Fragment Object" width:140 align:#center enabled:false
        button generateForm_btn "Generate Form" width:140 align:#center enabled:false
        button generatePebbles_btn "Generate Pebbles" width:140 align:#center enabled:false

        on selectObject picked obj do
          (
            selectObject_tf.text = obj.name
            randomEdges_btn.enabled = true
            nbParts_spinner.enabled = true
            fragmentation_btn.enabled = true
          )


        on randomEdges_btn pressed do
          (
            myFunc.removeRandomEdges()
          )

        on nbParts_spinner changed amt do
          (
            nbParts = amt
          )

        on fragmentation_btn pressed do
          (
            myFunc.fragmentation()
            generateForm_btn.enabled = true
          )

        on generateForm_btn pressed do
          (
                myFunc.generateForm()
                generatePebbles_btn.enabled = true
            )

        on generatePebbles_btn pressed do
            (
                myFunc.generatePebbles()
            )
    )

    createDialog generateBox 200 200 --create a dialog with the rollout
```

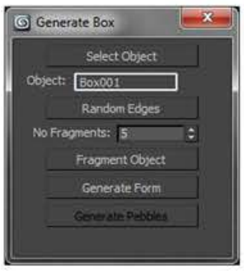
END Script



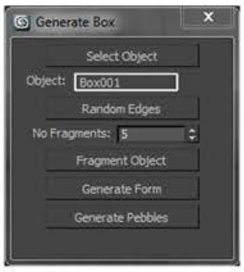
Object: NONE
No Fragments: 5



Object: Box001
No Fragments: 5



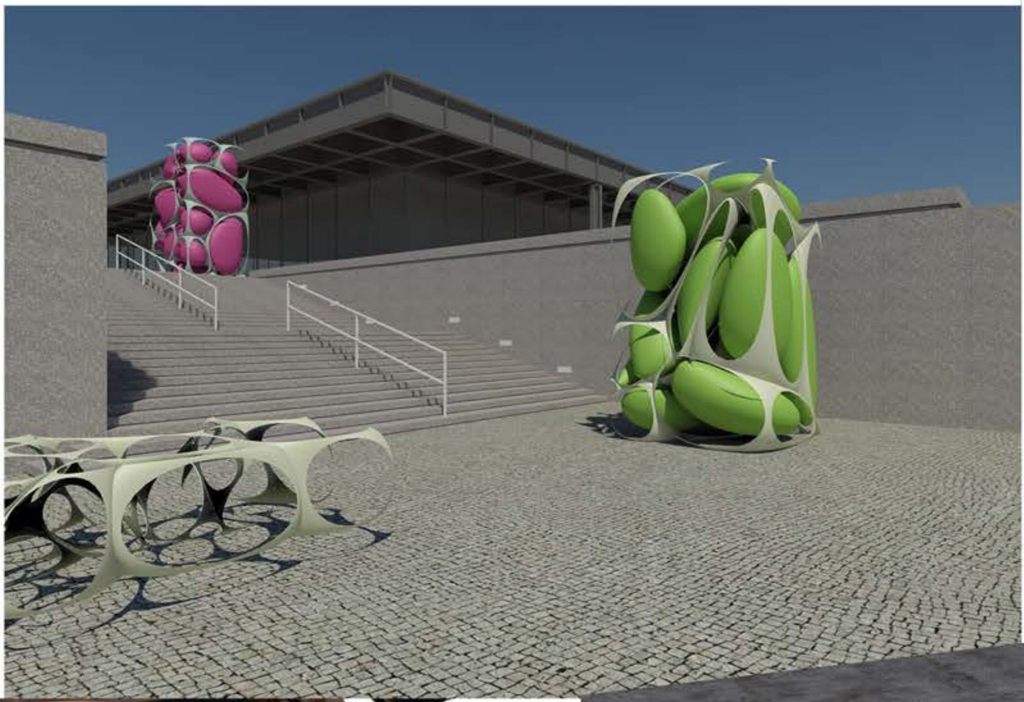
Object: Box001
No Fragments: 5

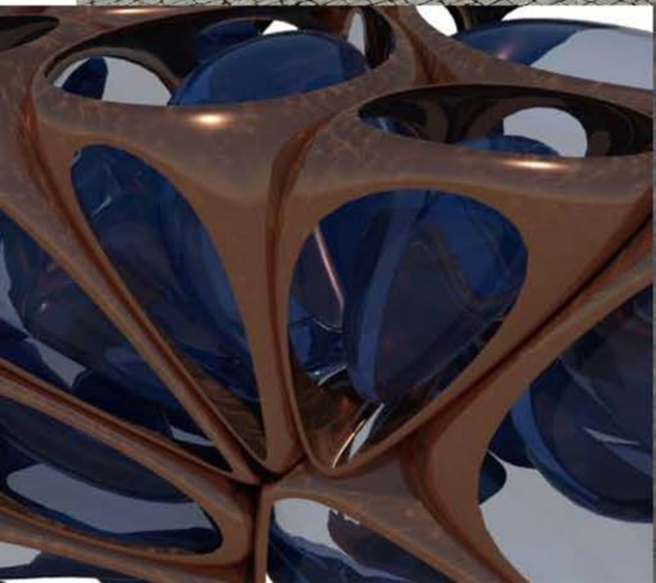


Object: Box001
No Fragments: 5

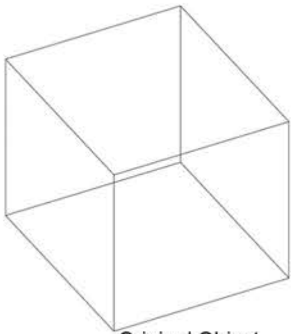
The way that the script runs is straight forward. The pick button has been made so that until an object has been selected the
-Random Edges
-No Fragments spinner   (we set the maximum fragments to 20 as it seems that above this number 3DsMax seems to have some difficulty generating the Fragments)
-Fragment Object
buttons are disabled.The generate form button is also disabled until the object has been fragmented and likewise only when the Generate Form button has been pressed can you Generate Pebbles.

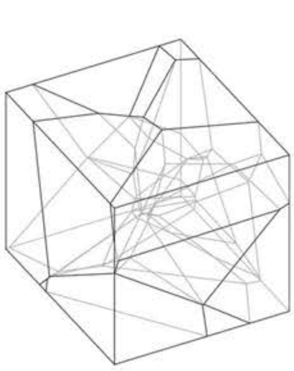
Renders

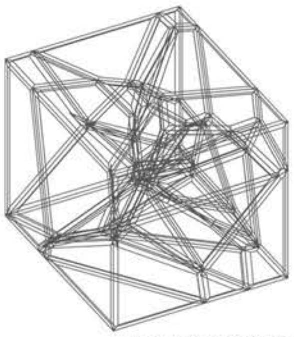


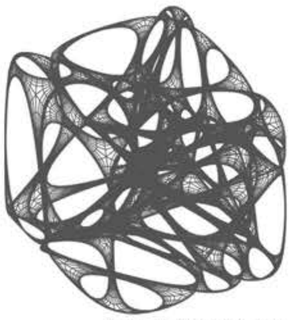


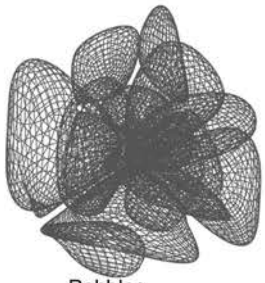
Architectural Applications of Script

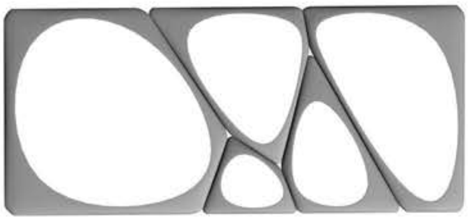

Original Object

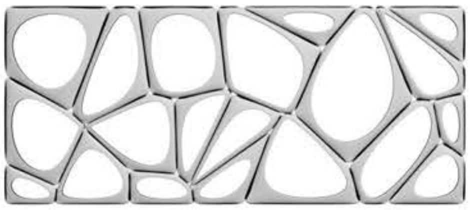

Fragment


Inset + Deleted Faces

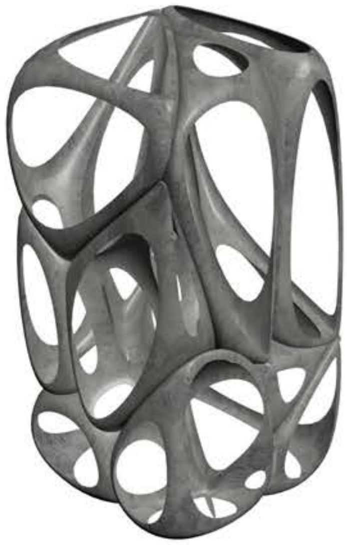

Nurms Subdivision

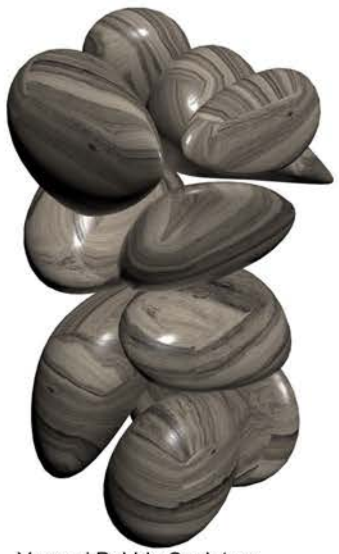

Pebbles

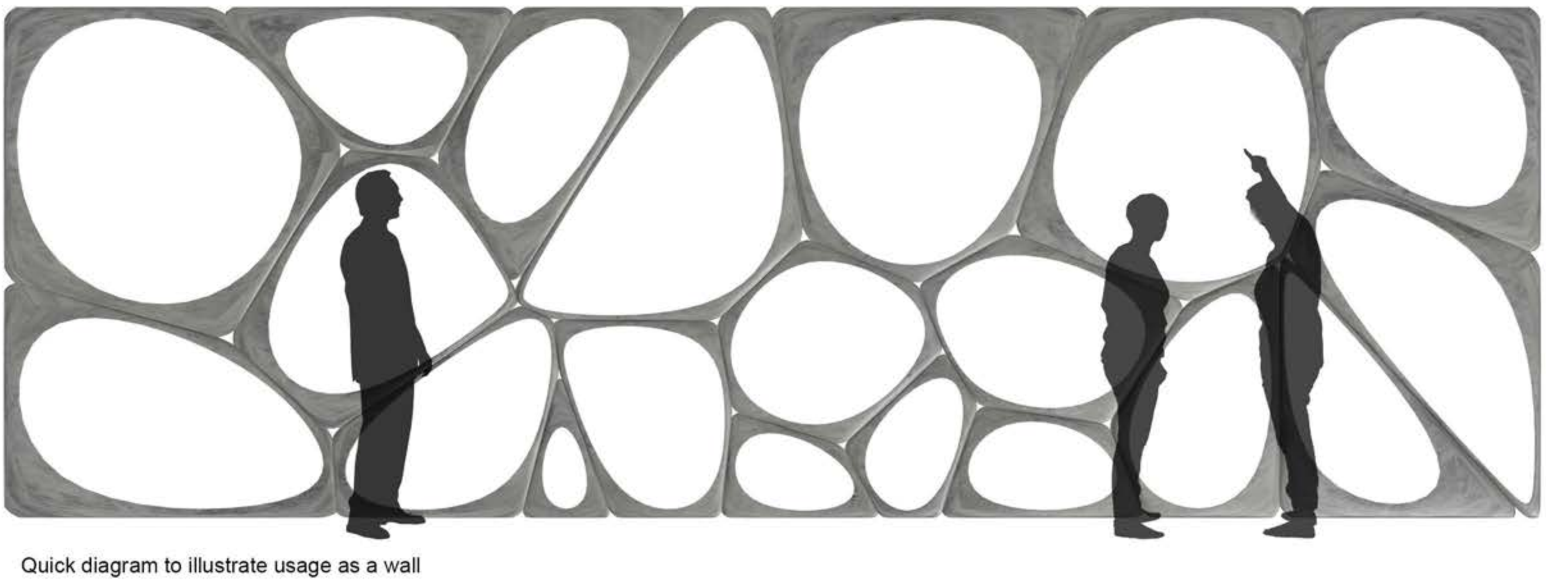




The script has several architectural applications for instance it could be used to create a voronoi frame to be used as a building envelope. Or it could be used as an decorative interior wall. With the scripts potential to be used in many different ways, we believe that we have reached a good end result. We see the possibility to continue and use this script for future design projects


Voronoi Bone Sculpture


Voronoi Pebble Sculpture


Quick diagram to illustrate usage as a wall