|  |
| --- |
| **Splitting Cells (Python-Rhino)** |

**import** rhinoscriptsyntax **as** rs

**import** random

cellsplit=3

startcells =rs.**GetObjects**("Startcells", filter=0,group=**True**)

cellsize=2

VolumeSize =rs.**GetInteger**("Size of volume in m3")

VolumofOneSphere=33.6

size=2.89

size=VolumeSize/VolumofOneSphere

**def Split**(StartCells,CellSplit):

 Amount=**len**(StartCells)

 EmptyList=[]

 NewPoints=[]

 **for** s **in** rs.**frange**(0**,**(Amount-1),1):

 AffectingCell=StartCells[s]

 "make a random vector"

 x=random.**randrange**(-5, 5, 1)

 y=random.**randrange**(-5, 5, 1)

 z=random.**randrange**(-10, 20, 1)

 NullPoint=rs.**AddPoint**([0,0,0])

 MovedPoint=rs.**AddPoint**([x,y,z])

 MovedPoint2=rs.**AddPoint**([x,-y,z])

 MoveVector=rs.**VectorCreate**(MovedPoint,NullPoint)

 MoveVector2=rs.**VectorCreate**(MovedPoint2,NullPoint)

 MoveVectorUni=rs.**VectorUnitize**(MoveVector)

 MoveVectorUni2=rs.**VectorUnitize**(MoveVector2)

 "Make Points for cells"

 Point1=rs.**AddPoint**(AffectingCell)

 Point2=rs.**AddPoint**(AffectingCell)

 "MoveObjects"

 rs.**MoveObject**(Point1,cellsize\*MoveVectorUni)

 rs.**MoveObject**(Point2,cellsize\*MoveVectorUni2)

 "Delete The vectorcreators"

 rs.**DeleteObject**(NullPoint)

 rs.**DeleteObject**(MovedPoint)

 rs.**DeleteObject**(MovedPoint2)

 "Test if the point is not conflicting"

 "Make two Lists"

 distancesPoint1=[]

 distancesPoint2=[]

 "All Points in the field where it can possible conflict with"

 AllPointStartCells=StartCells

 MeanTimeadded=[]

 AllPoints=AllPointStartCells+MeanTimeadded

 Amount2=**len**(AllPoints)

 **for** l **in** rs.**frange**(0**,**(Amount2-1),1):

 distancesPoint1toAllPoint=rs.**Distance**(Point1,AllPoints[l])

 distancesPoint1.**append**(distancesPoint1toAllPoint)

 distancesPoint2toAllPoint=rs.**Distance**(Point2,AllPoints[l])

 distancesPoint2.**append**(distancesPoint2toAllPoint)

 list.**sort**(distancesPoint1)

 list.**sort**(distancesPoint2)

 **if** (distancesPoint1[0]<cellsize):

 rs.**DeleteObject**(Point1)

 **else**:

 NewPoints.**append**(Point1)

 **if** (distancesPoint2[0]<cellsize):

 rs.**DeleteObject**(Point2)

 **else**:

 NewPoints.**append**(Point2)

 **if** (distancesPoint1[0]>cellsize) **and** (distancesPoint2[0]>cellsize) :

 **print** "There has been a double split"

 **if** (distancesPoint1[0]<cellsize) **and** (distancesPoint2[0]>cellsize) :

 **print** "There has been a one split"

 **if** (distancesPoint1[0]>cellsize) **and** (distancesPoint2[0]<cellsize) :

 **print** "There has been a one split"

 **if** (distancesPoint1[0]<cellsize) **and** (distancesPoint2[0]<cellsize) :

 **print** "No Split for me in this round"

 AmountofNewPointsCreation=**len**(NewPoints)

 StartCells=StartCells+NewPoints

 **if len**(StartCells)<size:

 **Split**(StartCells,CellSplit)

 **else**:

 SpheresAmount=**len**(StartCells)

 **for** p **in** rs.**frange**(0**,**(SpheresAmount-1),1):

 q=rs.**AddPoint**(StartCells[p])

 rs.**AddSphere**(q,cellsize\*0.5)

**Split**(startcells,cellsplit)

|  |
| --- |
| **Splitting Cells (Python-Rhino) END** |

|  |
| --- |
| **Network Movement (Python-Rhino)** |

**import** rhinoscriptsyntax **as** rs

**import** time

targets=rs.**GetPoints**(**True**,**False**,"Select the entrances", "go on!", 20, **None**)

targetsoldpos=rs.**GetPoints**(**True**,**False**,"Select where it came from", "go on!", 20, **None**)

amountoftargets=**len**(targets)

affection=rs.**GetReal**('BendingFactor low means easy bending',0.80,0.70,0.99)

walksteps=150

start=rs.**GetPoint**("Select start point")

Direction1=rs.**GetPoint**("Select Direction point")

Startvector=rs.**VectorCreate**(Direction1,start)

rs.**VectorUnitize**(Startvector)

Startvector=Startvector

Endvector=[]

**for** b **in** rs.**frange**(0,amountoftargets-1,1):

 endelvec=rs.**VectorCreate**(targets[b],targetsoldpos[b])

 rs.**VectorUnitize**(endelvec)

 Endvector.**append**(0.1\*endelvec)

myListofCurrentVectors=[]

**for** m **in** rs.**frange**(0,amountoftargets-1,1):

 myListofCurrentVectors.**append**(0.1\*Startvector)

myListofPeople=[]

**for** j **in** rs.**frange**(0**,**(amountoftargets-1),1):

 Walk=rs.**AddPoint**(0,0,0)

 rs.**MoveObject**(Walk,start)

 myListofPeople.**append**(Walk)

EndPoints=[]

**for** p **in** rs.**frange**(0**,**(amountoftargets-1),1):

 Epo=rs.**AddPoint**(0,0,0)

 rs.**MoveObject**(Epo,targets[p])

 EndPoints.**append**(Epo)

**def walking**(amount, targs, people,steps,svector,evector,ePoints):

 myListofAimedVectors=[]

 myListofAimedVectors2=[]

 EmptyList=[]

 ListofLines=[]

 **for** k **in** rs.**frange**(0**,**(amount-1),1):

 linepoints=[]

 linepoints2=[]

 veccies=rs.**VectorCreate**(ePoints[k],people[k])

 veccies2=rs.**VectorCreate**(people[k],ePoints[k])

 univeccies=rs.**VectorUnitize**(veccies)

 univeccies2=rs.**VectorUnitize**(veccies2)

 myListofAimedVectors.**append**(univeccies)

 myListofAimedVectors2.**append**(univeccies2)

 svector[k]=affection\*svector[k]

 evector[k]=affection\*evector[k]

 myListofAimedVectors[k]=0.05\*myListofAimedVectors[k]

 myListofAimedVectors2[k]=0.05\*myListofAimedVectors2[k]

 svector[k]=svector[k]+myListofAimedVectors[k]

 evector[k]=evector[k]+myListofAimedVectors2[k]

 epointbeforemove=ePoints[k]

 peoplebeforemove=people[k]

 rs.**MoveObjects**(ePoints[k],evector[k])

 rs.**MoveObjects**(people[k],svector[k])

 **if** rs.**Distance**(ePoints[k],people[k])>2:

 epointaftermove=ePoints[k]

 peopleaftermove=people[k]

 linepoints.**append**(epointbeforemove)

 linepoints.**append**(epointaftermove)

 rs.**AddCurve**(linepoints)

 linepoints2.**append**(peoplebeforemove)

 linepoints2.**append**(peopleaftermove)

 rs.**AddCurve**(linepoints2)

 linepoints=EmptyList

 linepoints2=EmptyList

 myListofAimedVectors=EmptyList

 myListofAimedVectors2=EmptyList

 rs.**Sleep**(500)

 ListofPoints=EmptyList

 **if** steps >=1:

 **walking**(amount,targs,people,steps-1,svector,evector,ePoints)

**walking**(amountoftargets, targets, myListofPeople,walksteps, myListofCurrentVectors, Endvector,EndPoints)

|  |
| --- |
| **Network Movement (Python-Rhino) END** |

|  |
| --- |
| **Control and Export Wind (Python-Maya)****First time running use this script, then change:**def Send( \*pArgs ):**Into:****def cancelCallback( \*pArgs ):****Matching files in folder Maya To Rhino ( Grasshopper & Maya )** |

# keyRotationWithUI.py

import maya.cmds as cmds

import functools

import socket

def createUI( pWindowTitle, pApplyCallback ):

 windowID = 'myWindowID'

 if cmds.window( windowID, exists=True ):

 cmds.deleteUI( windowID )

 cmds.window( windowID, title=pWindowTitle, sizeable=False, resizeToFitChildren=True )

 cmds.rowColumnLayout( numberOfColumns=3, columnWidth=[ (1,200), (2,60), (3,60) ], columnOffset=[ (1,'right',3) ] )

 cmds.text( label='Wind Direction (0=north,90=west):' )

 windDirField = cmds.intField( value=cmds.playbackOptions( q=True, minTime=True ) )

 cmds.text( label='degrees' )

 cmds.text( label='Wind Speed' )

 windSpeedField = cmds.intField(v=80 )

 cmds.text( label='m/s' )

 cmds.text( label='Particle/second:' )

 partSecField = cmds.intField(v=50)

 cmds.separator( h=10, style='none' )

 def Send( \*pArgs ):

 if cmds.window( windowID, exists=True ):

 cmds.deleteUI( windowID )

 cmds.button( label='Send Locations', command=Send )

 cmds.button( label='Apply', command=functools.partial( pApplyCallback,

 windDirField,

 windSpeedField,

 partSecField ) )

 def cancelCallback( \*pArgs ):

 if cmds.window( windowID, exists=True ):

 cmds.deleteUI( windowID )

 cmds.button( label='Cancel', command=cancelCallback )

 cmds.separator( h=10, style='none' )

 cmds.separator( h=10, style='none' )

 cmds.separator( h=10, style='none' )

 cmds.button( label='Play', command=Send )

 cmds.button( label='Pauze', command=Send )

 cmds.button( label='Stop', command=Send )

 cmds.showWindow()

def applyCallback( pWindDirField, pWindSpeedField, pPartSecField, \*pArgs ):

 # print 'Apply button pressed.'

 windD = cmds.intField( pWindDirField, query=True, value=True )

 speed = cmds.intField( pWindSpeedField, query=True, value=True )

 partsecond = cmds.intField( pPartSecField, query=True, value=True )

 cmds.currentTime( 0 )

 print 'WindDirection: %s' % ( windD )

 print 'Speed: %s' % ( speed )

 print 'Particles/sec: %s' % ( partsecond )

 cmds.select('pPlane1')

 cmds.rotate(180,0,(windD+255))

 cmds.select('locator1')

 cmds.rotate(180,0,(windD+165))

 selectionList = cmds.ls( selection=True, type='transform' )

 cmds.select('emitter3')

 cmds.setAttr("emitter3.rate", (partsecond))

 cmds.select('gravityField1')

 cmds.setAttr("gravityField1.magnitude", (speed))

 cmds.select('Killfield')

 cmds.rotate(180,0,(windD-15))

createUI( 'WindController', applyCallback )

def Send(pShowTime):

 UDP\_IP1="127.0.0.1"

 UDP\_IP2="127.0.0.2"

 UDP\_IP3="127.0.0.3"

 UDP\_IP4="127.0.0.4"

 UDP\_IP5="127.0.0.5"

 UDP\_IP6="127.0.0.6"

 UDP\_IP7="127.0.0.7"

 UDP\_IP8="127.0.0.8"

 UDP\_IP9="127.0.0.9"

 UDP\_IPTIME="127.127.127.127"

 UDP\_PORT1=7001

 UDP\_PORT2=7002

 UDP\_PORT3=7003

 UDP\_PORT4=7004

 UDP\_PORT5=7005

 UDP\_PORT6=7006

 UDP\_PORT7=7007

 UDP\_PORT8=7008

 UDP\_PORT9=7009

 UDP\_PORTTIME=127

 cmds.select ( 'nParticle4')

 dens = cmds.getAttr('nParticle4.position')

 Amount=len(dens)

 MESSAGETIME=str(cmds.currentTime( query=True ))

 print MESSAGETIME

 MESSAGE1=str(dens[0:400])

 MESSAGE2=str(dens[400:800])

 MESSAGE3=str(dens[800:1200])

 MESSAGE4=str(dens[1200:1600])

 MESSAGE5=str(dens[1600:2000])

 MESSAGE6=str(dens[2000:2400])

 MESSAGE7=str(dens[2400:2800])

 MESSAGE8=str(dens[2800:3200])

 MESSAGE9=str(dens[3200:3600])

 MESSAGECONTROL=str("1234")

 UDP\_IPCON="1.2.3.4"

 UDP\_PORTCON=1234

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGECONTROL, (UDP\_IPCON, UDP\_PORTCON) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGETIME, (UDP\_IPTIME, UDP\_PORTTIME) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE1, (UDP\_IP1, UDP\_PORT1) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE2, (UDP\_IP2, UDP\_PORT2) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE3, (UDP\_IP3, UDP\_PORT3) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE4, (UDP\_IP4, UDP\_PORT4) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE5, (UDP\_IP5, UDP\_PORT5) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE6, (UDP\_IP6, UDP\_PORT6) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE7, (UDP\_IP7, UDP\_PORT7) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE8, (UDP\_IP8, UDP\_PORT8) )

 sock.close()

 sock = socket.socket( socket.AF\_INET,socket.SOCK\_DGRAM )

 sock.sendto( MESSAGE9, (UDP\_IP9, UDP\_PORT9) )

 sock.close()

 print Amount

|  |
| --- |
| **Control and Export Wind (Python-Maya) END** |