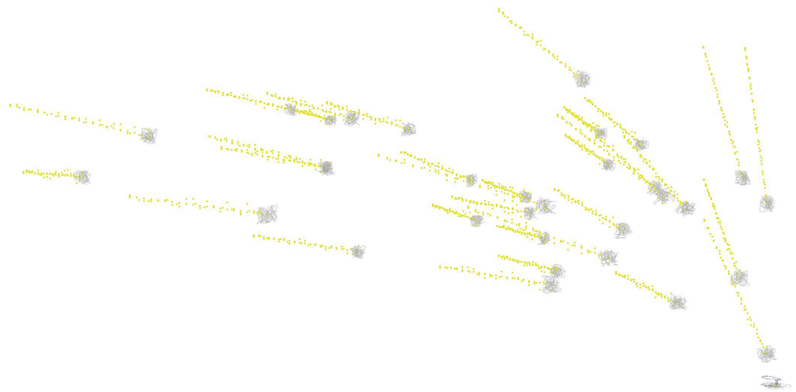
Final Project DM2

Homage to Urania - Meteorites over Vienna

The person

Urania is the Muse of Astronomy in Greek mythology, her name derived from Uranos, the personification of the sky. She is often depicted with symbols such as a celestial globe and a pointer, which she typically holds in her hands, sometimes gesturing with the pointer towards a globe on the ground.
In Late Antiquity, the Pythagorean concept of the harmony of the spheres was elaborated upon, with each Muse assigned to a specific celestial sphere. In this framework, Urania was associated with the fixed stars, representing the highest tone in the harmony of the spheres.



The project

As the foundation for my project, I used the "Urania" Observatory in Vienna. Named after Urania, the Muse of Astronomy and the stars, the observatory is one of the oldest public observatories in Austria and embodies the spirit of exploration and the quest for knowledge about the cosmos.
In my project, I envisioned meteorites descending upon the observatory, each following different celestial paths. As they shower down upon the building, they create a stunning spectacle, illuminating the sky and cover the building in the process.
This project pays homage to Urania, not only as a symbol of celestial navigation and discovery but also as a representation of humanity's enduring fascination with the stars. The meteoric paths are a reminder of the dynamic nature of the universe, where cosmic events continuously unfold, shaping our understanding of space and time.

The code

```
allBahnCoords = []
anzMet = random.randint(20, 40)
anzFrames = 100
captureIt = 0

deltaZ = dZ = 5/anzMet
divY = 20
divX = 15
min_distance = 5
max_distance = 25

def MakeMeteo(metPt, mSize=25):
    meteoritCoords = []
    for i in range(80):
        vecX = dm.randVec(-mSize, mSize)
        corX = rs.VectorAdd(metPt, vecX)
        meteoritCoords.append(corX)
    meteorit = rs.AddCurve(meteoritCoords, 2)
    rs.ObjectColor(meteorit, [102,102,102])
    rs.ObjectName(meteorit, "meteo_meteorit_"+str(m))

    ###Meteo flachdruecken
    pos = rs.CurveStartPoint(meteorit)
    if pos[2] <= 20.5:
        stauchung = random.uniform(0.1,0.5)
        rs.ScaleObject(meteorit,metPt, [1.5, 1.5, stauchung])
    ###Tangente auf Bahn
    for bahn in rs.ObjectsByName("bahn_*"):
        #print bahn
        param = rs.CurveClosestPoint(bahn, metPt)
        ptBahn = rs.EvaluateCurve(bahn, param )
        dist = rs.Distance( metPt, ptBahn )
        #print dist
        if dist < 0.00001:
            tan = rs.CurveTangent(bahn, param)
            schweif = rs.AddLine( metPt, rs.VectorAdd(metPt, rs.VectorScale(tan,
-random.uniform(200,500))) )
     ###Koma
            coordsKoma = rs.DivideCurve(schweif, divY, 0)
            #rs.DeleteObject(schweif)
            SchweifPt = rs.AddPoints(coordsKoma)
            rs.ObjectName(SchweifPt, "schweif_"+str(m))
            rs.ObjectColor(SchweifPt, [200,200,0])
            for i in range(len(coordsKoma)):
                auskrag = dm.reMap(i, 0, len(coordsKoma)-1, max_distance, min_distance)
                KomaVec = rs.VectorAdd(coordsKoma[i], [0,0, random.uniform(-auskrag, auskrag)])
                Koma = rs.AddPoint(KomaVec)
                rs.ObjectName(Koma, "koma_"+str(m))
                rs.ObjectColor(Koma, [200,200,0])
    ###zusaetzliche Punkte um Koma auszufuellen
            coordsKomaZusatz = rs.DivideCurve(schweif, divX, 0)
            rs.DeleteObject(schweif)
            for i in range(len(coordsKomaZusatz)):
                auskragAdd = dm.reMap(i, 0, len(coordsKomaZusatz)-1, max_distance, min_distance)
                KomaVecAdd = rs.VectorAdd(coordsKomaZusatz[i], [0,0, random.uniform(-auskrag, auskrag)])
                KomaAdd = rs.AddPoint(KomaVecAdd)
                rs.ObjectName(KomaAdd, "koma_add_"+str(m))
                rs.ObjectColor(KomaAdd, [200,200,0])
            #break

for i in range(anzMet):
    tar = random.choice(targets)
    tar[2]= i*dZ
    #rs.AddPoint( tar )
    pOben = rs.VectorAdd(tar, [0,0, random.uniform(200,500)] )
    pStart= rs.VectorAdd(pOben,
[random.uniform(-100,2000),random.uniform(-100,2000),random.uniform(100,500)] )
    kometBahnCoords = [pStart, pOben, tar ]
    bahn = rs.AddCurve( kometBahnCoords, 2) ###Flugbahn
    rs.HideObject(bahn)
    rs.ObjectColor(bahn, [220,220,100])
    rs.ObjectName(bahn, "bahn_"+str(i))
    anzBahn = int(rs.CurveLength(bahn)*random.uniform(0.01, 0.05)) ### abstand der points =>
(randomisierte) speed
    #print "anzBahn", anzBahn
    coords = rs.DivideCurve( bahn, anzBahn, 0)
    for x in range(random.randint(1, 3)): ## sooft startet neuer komet auf gleicher bahn == maximale
dauer meteoritenRegen .. nicht mehr als 8 einstellen :)
        dm.esc()
        coords.extend(coords)
    allBahnCoords.append( coords )
    #rs.DeleteObject(bahn)

for j in range(anzFrames):
    #zoom+=deltaZoom
    #camVec = rs.VectorRotate(camVec, deltaAngle, [0,0,1])
    #cam = rs.VectorAdd(tar, camVec)
    #dm.setCameraTarget(cam, tar, lens= zoom, rota= 0, upVec =
[0.02312681190534399,0.04104226012737455,0.9988897253724917])
    rs.Redraw()
    dm.esc()
    #rs.DeleteObjects(rs.ObjectsByName("meteo_*"))
    rs.DeleteObjects(rs.ObjectsByName("schweif_*"))
    rs.DeleteObjects(rs.ObjectsByName("koma_*"))
    meteos = rs.ObjectsByName("meteo_*")
    for met in meteos:
        pos = rs.CurveStartPoint(met) ###rs.PointCoordinates( met )
        if pos[2] > 20.5: ### unterhalb bleibens liegen
            rs.DeleteObject( met )

    for m in range(anzMet):
        dm.esc()
        if j < len(allBahnCoords[m]):
            metPt = rs.AddPoint(allBahnCoords[m][j])
            MakeMeteo(metPt)
            rs.DeleteObject(metPt)
            #rs.ObjectName(rs.AllObjects()[0], "meteo_"+str(m))
```