BeamDistribution.fly2

```
function IonType (name, mass, charge, fraction)
    return {
        name = name,
        mass = mass,
        charge = charge,
        fraction = fraction
    }
end

local spawnregion_length = 22 -- mm
-- fractIons from straub et al.
local argon_ions = {
    IonType('Ar+', 39.948, 1, 2.64),
    IonType('Ar2+', 39.948, 2, 0.189)
}

local background_ions = {
    IonType('CO2+', 43, 1, 0.02),
    IonType('O2+', 32, 1, 0.02),
    IonType('N2+', 28, 1, 0.05),
    IonType('H2O+', 18, 1, 1.01),
    IonType('OH+', 17, 1, 0.3),
    IonType('O+', 16, 1, 0.07),
    IonType('C+', 12, 1, 0.01),
    IonType('H2+', 2, 1, 0.024),
    IonType('H+', 1, 1, 0.19)
}

local test_ions = {}

for i = 1, 30 do
    test_ions[#test_ions+1] = IonType('Test'..i, i, 1, 1 / 30)
end

local ion_types = argon_ions
local amount_multiplier = 3000
local beam_center = 11 -- mm
local fwhm = 1.95 -- mm
local ion_kinetic_energy = 1/25 -- eV

-- build the particle table type by type
local t = {coordinates = 0}

for k, ion_type in ipairs(ion_types) do
    local num_ions = math.ceil(amount_multiplier * ion_type.fraction)
    t[#t+1] = standard_beam {
        n = num_ions,
        tob = 0,
        mass = ion_type.mass,
        charge = ion_type.charge,
        x = gaussian_distribution {
          mean = beam_center,
          fwhm = fwhm
        },
        y = uniform_distribution {
          min = -spawnregion_length/2,
          max = spawnregion_length/2
        },
        z = gaussian_distribution {
            mean = 0,
            fwhm = fwhm
        },
        ke = ion_kinetic_energy,
        cwf = 1,
        color = k,
        direction = cone_direction_distribution {
          axis = vector(1, 0, 0),
          half_angle = 180, -- 360 degrees - random
          fill = true
        }
    }
end

particles(t)

```