```
# === JANUS 100 MILLIONS – GRATUIT POUR TOUS ===
import numpy as np, matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation
import time

print("Lancement run 100M particules Janus – GRATUIT – 06/11/2025 23h25 CET")

# Paramètres
N = 100_000_000 # 100 MILLIONS
box = 500.0 # Mpc
G = 1.0
dt = 0.0005
steps = 500
soft = 0.05

# Initialisation (optimisée mémoire)
pos = np.random.uniform(-box/2, box/2, (N, 3)).astype(np.float32)
vel = np.zeros((N, 3), dtype=np.float32)
masses = np.random.choice([1.0, -1.0], N).astype(np.float32)

# Force bimétrique (tree approx via grid 64³)
grid_size = 64
cell_size = box / grid_size
density = np.zeros((grid_size,)*3, dtype=np.float32)

def update_density():
  global density
```

```
    density.fill(0)
    idx = np.floor((pos + box/2) / cell_size).astype(int) % grid_size
    np.add.at(density, (idx[:,0], idx[:,1], idx[:,2]), masses)

def force_from_grid():
    acc = np.zeros_like(pos)
    for i in range(N):
        ix, iy, iz = np.floor((pos[i] + box/2) / cell_size).astype(int) % grid_size
        for dx in [-1,0,1]:
            for dy in [-1,0,1]:
                for dz in [-1,0,1]:
                    nx, ny, nz = (ix+dx)%grid_size, (iy+dy)%grid_size, (iz+dz)%grid_size
                    r = pos[i] - np.array([nx,ny,nz])*cell_size + cell_size/2
                    d2 = np.sum(r**2) + soft**2
                    acc[i] += G * density[nx,ny,nz] * r / d2**1.5 * masses[i]
    return acc

# Simulation
start = time.time()
traj = []
for step in range(steps):
    update_density()
    acc = force_from_grid()
    vel += acc * dt
    pos += vel * dt
    if step % 50 == 0:
        traj.append(pos[::5000].copy()) # Sous-échantillon pour GIF
    if step % 100 == 0:
```

```
    print(f"Step {step}/{steps} – t = {step*dt*13.8/0.5:.1f} Gyr")

# GIF (sous-échantillon 20k)
fig = plt.figure(figsize=(8,8)); ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter([],[],[], c=[], cmap='bwr', s=1)
ax.set
```