**MBT Sheet Transition Simulation Replication Kit**

Purpose:

Simulates dynamic MBT quantum sheets as evolving wave patterns. Detects high intensity “holes” in one sheet and triggers correlated “jump” flashes in other sheets, modelling quantum sheet transitions such as wormholes or black hole bounces.

**Code**

import numpy as np

import matplotlib.pyplot as plt

import imageio.v2 as imageio

import os

# PARAMETERS

num\_sheets = 3

grid\_size = 60

frames = 90

hole\_threshold = 0.91  # percentile for "hole" detection (between 0 and 1)

jump\_flash\_duration = 6  # frames to keep the flash alive

np.random.seed(42)  # For reproducibility

# CREATE DYNAMIC OCEAN SHEETS (rolling MBT waves)

def make\_sheet\_waves(seed\_offset=0):

   np.random.seed(100 + seed\_offset)

   waves = []

   phase = np.random.uniform(0, 2\*np.pi, (3,))

   amp = np.random.uniform(0.15, 0.35, (3,))

   freq = np.random.uniform(1.1, 2.2, (3,))

   base = np.linspace(-1, 1, grid\_size)

   X, Y = np.meshgrid(base, base)

   for t in range(frames):

       val = (

           amp[0] \* np.sin(freq[0]\*X + t\*0.10 + phase[0]) +

           amp[1] \* np.cos(freq[1]\*Y + t\*0.07 + phase[1]) +

           amp[2] \* np.sin(freq[2]\*(X+Y) + t\*0.05 + phase[2])

       )

       val += np.random.normal(scale=0.03, size=val.shape)

       waves.append(val)

   return np.stack(waves)

# Generate three sheets

sheets = [make\_sheet\_waves(i) for i in range(num\_sheets)]

# Find holes per sheet/frame

holes = []

for s in range(num\_sheets):

   mask = sheets[s] > np.quantile(sheets[s], hole\_threshold)

   holes.append(mask)

# Initialize jump/flash memory

flashes = np.zeros((frames, num\_sheets, grid\_size, grid\_size), dtype=int)

# Whenever a hole forms in one sheet, instantly flash a jump in a random other sheet

for t in range(frames):

   for s in range(num\_sheets):

       coords = np.argwhere(holes[s][t])

       for (i, j) in coords:

           # Pick a different sheet to flash

           other\_sheets = [x for x in range(num\_sheets) if x != s]

           flash\_s = np.random.choice(other\_sheets)

           # Flash at same (i,j), for jump\_flash\_duration frames

           for dt in range(jump\_flash\_duration):

               tt = min(t+dt, frames-1)

               flashes[tt, flash\_s, i, j] = 1

# OUTPUT FOLDER

output\_dir = "/mnt/data/mbt\_quantum\_ocean\_frames"

os.makedirs(output\_dir, exist\_ok=True)

# PLOTTING AND FRAME GENERATION

filenames = []

for t in range(frames):

   fig, axes = plt.subplots(1, num\_sheets, figsize=(13, 4))

   for s in range(num\_sheets):

       ax = axes[s]

       im = ax.imshow(sheets[s][t], cmap="ocean", vmin=-0.8, vmax=0.8)

       # Holes (white)

       ax.contour(holes[s][t], levels=[0.5], colors='white', linewidths=1.7)

       # Flashes (red)

       flash\_idx = np.where(flashes[t, s] > 0)

       ax.scatter(flash\_idx[1], flash\_idx[0], color='red', s=14, marker='o', label="Jump/Flash")

       ax.set\_title(f"Sheet {s+1}")

       ax.set\_xticks([]), ax.set\_yticks([])

   plt.suptitle("Living MBT Quantum Ocean Sheets\n(White = Hole; Red = Jump)")

   plt.tight\_layout(rect=[0, 0, 1, 0.95])

   fname = f"{output\_dir}/frame\_{t:03d}.png"

   plt.savefig(fname, dpi=120)

   plt.close()

   filenames.append(fname)

# SAVE AS VIDEO (MP4)

video\_path = "/mnt/data/mbt\_quantum\_ocean\_sheets.mp4"

with imageio.get\_writer(video\_path, fps=10) as writer:

   for fname in filenames:

       image = imageio.imread(fname)

       writer.append\_data(image)

print(f"Animation saved as: {video\_path}")

**Notes:**

* The code generates wave-like dynamic MBT sheets over time.
* Holes (high-value points above the threshold) are detected per sheet and frame.
* A hole triggers a jump/flash in a different sheet at the same coordinates for several frames, simulating quantum sheet transitions.
* The output is a saved mp4 animation showing sheet evolution, holes (white contours), and flashes (red dots).
* No changes were made to your original logic or parameters.

If you want, I can help prepare a README or usage guide file to include with the kit. Otherwise, you’re good to package and distribute!

What should we do next?