

CONVENTIONS

dims=(nx,ny,nz)] · **negative dimension = wrap**
1D/2D/3D; omit unused axes. Negative sizes enable periodic boundaries.

hood = array of offset tuples
Neighborhoods contain relative (dx[,dy[,dz]]) offsets.

x... / X... / ltoX... / DispWrapX...
X shorthand includes corresponding Y/Z variants.

i = linear site index · **SQ = lattice site** · **a = agent ID**
Coordinates are integer sites or continuous positions.

grid[i] / **grid[x,y,z]** · **slices copy**
Index by site or coordinates; slicing returns a copy.

Add(...) then **Update()** · **Reset()**
Queued changes apply synchronously.

SHARED GRID API · **NewGrid(dims, dtype)**

dims=(nx,ny,nz); negative size = wrap
1–3 dimensions; unused axes omitted.

grid.xDim... · **nDims** · **wrapX...**
Axis sizes, dimensionality, periodicity.

len(grid) · **grid[i]** · **grid[x,y,z]** · **grid[...]** = value
Site count; read/write; slices copy.

grid.Tol(x,y,z) · **ltoX...(i)**
Coordinate ↔ linear index.

grid.lnWrapX...(x)
Wrap coordinate on periodic axis.

grid.Box(x1,x2,y1,y2,z1,z2)
Half-open rectangular iterator.

grid.Hood(hood,x,y,z,unroll=False)
Offset neighborhood at a site.

AGENT GRID · **NewAgentGrid(dims, props=0, stack=False)**

a=agents.NewAgentSQ(x,y,z) · **NewAgent(x,y,z)**
Create at lattice site / continuous point.

agents.MoveSQ(a,x,y,z) · **Move(a,x,y,z)**
Move by lattice / continuous position.

agents.l(a) · **XSQ...(a)** · **X...(a)**
Site index, lattice and real coordinates.

agents.Alive(a) · **Dispose(a)** · **GetPop()**
Life status, removal, count.

agents[a,p] · **agents.counts**
Agent properties; read-only occupancy.

agents.AgentsAt(x,y,z) · **LastAgent(x,y,z)**
Site occupants / last occupant.

agents.AgentsInRadius(rad,x,y,z,exclude=None)
Nearby agents; periodic-aware.

agents.All(shuffle=False)
Snapshot of living agent IDs.

agents.lnWrapSQ...(x) · **DispWrapX...(p1,p2)**
Wrap lattice coordinate; shortest displacement.

START / UTILITIES

pal.FastMode() · **pal.njit(cache=True)**
Select fast mode before object creation; compile model kernels.

pal.Seed(s) · **Random()** · **RandInt(n)**
Shared PAL RNG; seed for reproducibility.

pal.MooreHood(dim, excludeCenter=False) · **VonNeumannHood(...)** · **CircleHood(dim, rad,...)**
Relative neighbor offsets.

pal.RGB(r,g,b) · **GetR/G/B(c)** · **ColorScale(colors,value)**
Pack, extract, interpolate colors.

pal.AwaitWindows(timeout=30)
Wait for rendering windows.

PDE GRID · **NewPDEgrid(dims)**

field.Add(v,x,y,z) · **Update()** · **Reset()**
Queue field deltas, apply, clear.

field.SetTimeSpaceStep(dt,dx,dy=1,dz=1) · **Dt()** · **Dx...()**
Numerical spacing.

field.Diffusion(rate,BC...) · **DiffusionMask(rate,mask,...)**
Uniform / masked diffusion.

field.DiffusionField(rates,BC...) · **DiffusionInterfaces(rx,ry,rz,BC...)**
Cell / face coefficients.

field.DiffusionADI(rate,BC...) · **DiffusionRadialCircle/Sphere(rate,outerBC)**
Implicit / radial diffusion.

field.Advection(vx,vy,vz,BC...) · **AdvectionField(vx,vy,vz,BC...)**
Uniform / spatial velocity.

field.AdvectionInterfaces(vx,vy,vz,BC...)
Face-centered velocity.

BC... = xMinBC,xMaxBC,yMinBC,yMaxBC,zMinBC,zMaxBC
Optional face boundary conditions.

POP GRID · **NewPopGrid(dims, capacity=None)**

pop.Add(v,x,y,z) · **Update()** · **Reset()**
Queue count deltas, apply, clear.

pop.GetPop() · **All()**
Total population; occupied indices.

LISTS · **NewList()** / **SAMPLER** · **NewMultinomial()**

lst.Append(i) · **Clear()** · **Random()** · **Shuffle()**
Integer-list operations.

lst.All() · **Iter()** · **len(lst)** · **lst[i]**
Copy / no-copy traversal, indexing.

m.Binomial(n,p) · **Setup(n)** · **Sample(p)**
Binomial sampling.

PIX WINDOW · **StartPixWindow(nx,ny,scale=1,...)**

pix[x,y]=pal.RGB(r,g,b) · **pix[i]=color**
Draw packed RGB pixels.

pix.xDim · **yDim** · **Tol(x,y)** · **ltoX/Y(i)** · **len(pix)**
Buffer geometry.

OPENGL WINDOW · **StartOpenGLWindow(nx,ny,nz=None,...)**

draw.Circle(rad,color,x,y,z=0) · **Box(xLen,color,x,y,z=0,...)**
Add circles/spheres and boxes.

draw.BoxSQ(color,x,y,z=0) · **Line(width,color,x1,y1,x2,y2,...)**
Site boxes and segments.

draw.Clear() · **win.Clear()** · **win.Background(color)**
Reset primitives / scene / background.

win.Borders(width,color) · **win.Camera(x,y,z,yaw=None,pitch=None)**
Domain border and 3D camera.

SHARED WINDOW API

win.IsOpen() · **Update()** · **Save(path,block=False)** · **Close()**
Window lifecycle / image output.

win.StartGif(path,delay=100) · **AddGifFrame(block=False)** · **StopGif()**
GIF recording; OpenGL supports timeout options.