Simulator: Tensor-ALU Module

```
#procedure input: #balls (n); (kx, ky); ball radius (r)
device kernel():
    i,j=tile co-ordinate based on thread ids
    if not_halo:
        worklist <- worklist_new  #stage1
        balls <- balls_new
        for b in worklist(i,j):
            for n in eight neighbouring_tiles of (i,j):
                if overlap(n,b):
                    if b is 0:
                        die()
                    naive_collider(n,b)  #updates position, velocity of 'balls'
    if first_thread_of_a_block:
        atomicInc(*counter,max_counter);  #init: counter=0, max_counter=total # blocks-1
    __syncthreads(); while(*counter);  #global barrier
    for b in worklist(i,j):  #stage2
        balls_new.position <- balls.position + (p==0)? keyboard_moves:balls.velocity
        balls_new.velocity <- balls.velocity
        reflect 'b' if it is at/outside boundary
        i_new, j_new <- new_tile_of_ball(b)
        atomically: worklist_new[i_new,j_new] = b

main():
    balls[0:n] <- initBalls()  #randomized positions, velocities
    glutInit(); glewInit()  #display, keyboard, timer, view-port inits
    while(True):
        if window_is_reshaped:
                tiles <- split window into rectangles based on (kx, ky, r)
                worklist <- assign balls to tiles based on their position
        device kernel(tiles assigned to GPU cores from threads->blocks in 2D) in stream1
        if approp. keyboard_input: convolution_kernel() in stream2 to modify background
        glutDraw(balls)
```

concepts

- Worklist
- Shared variables
- Structure of Arrays
- Pinned memory (cudaHostAlloc)
- Global barrier using voting and sync_threads()
- nvprof, nsys,
- Texture memory
- lock-step execution ternary operator