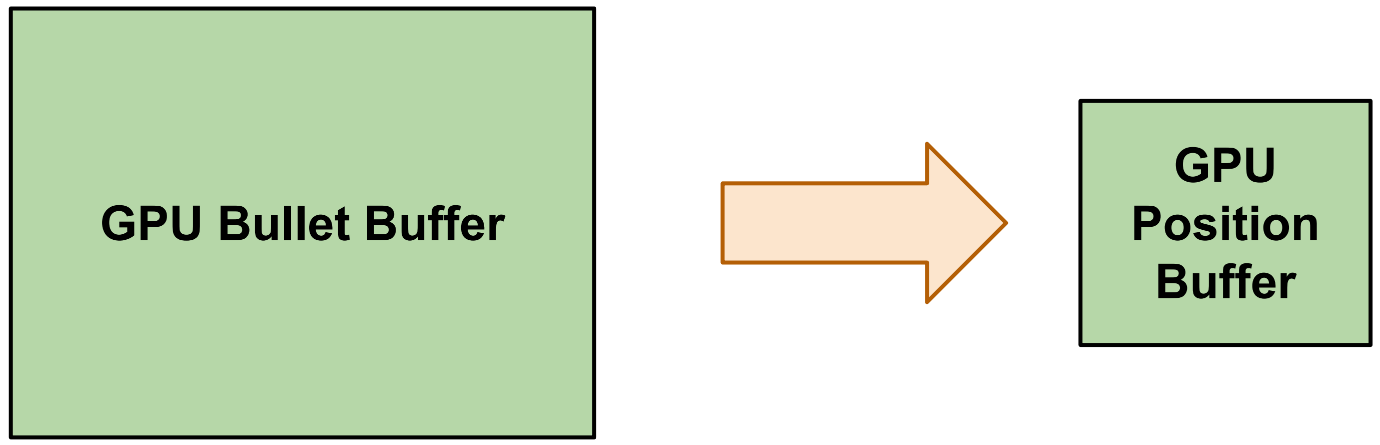
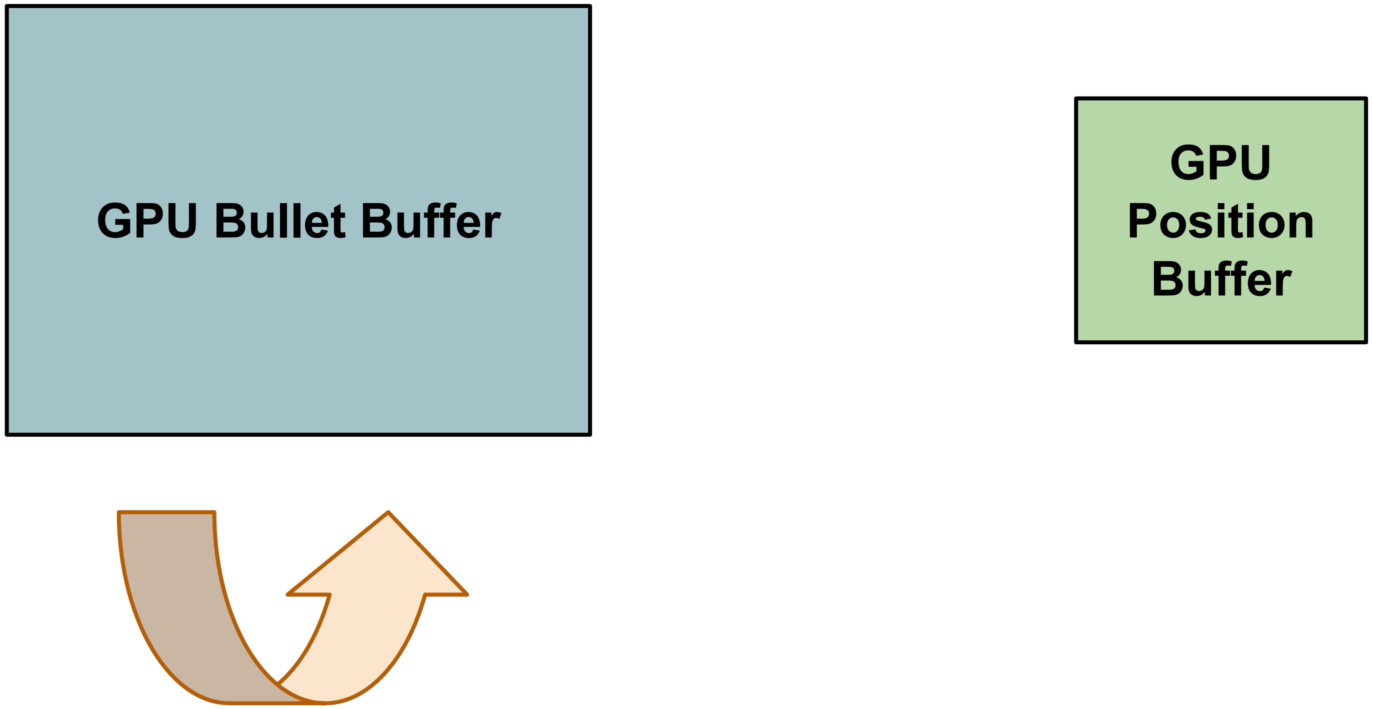
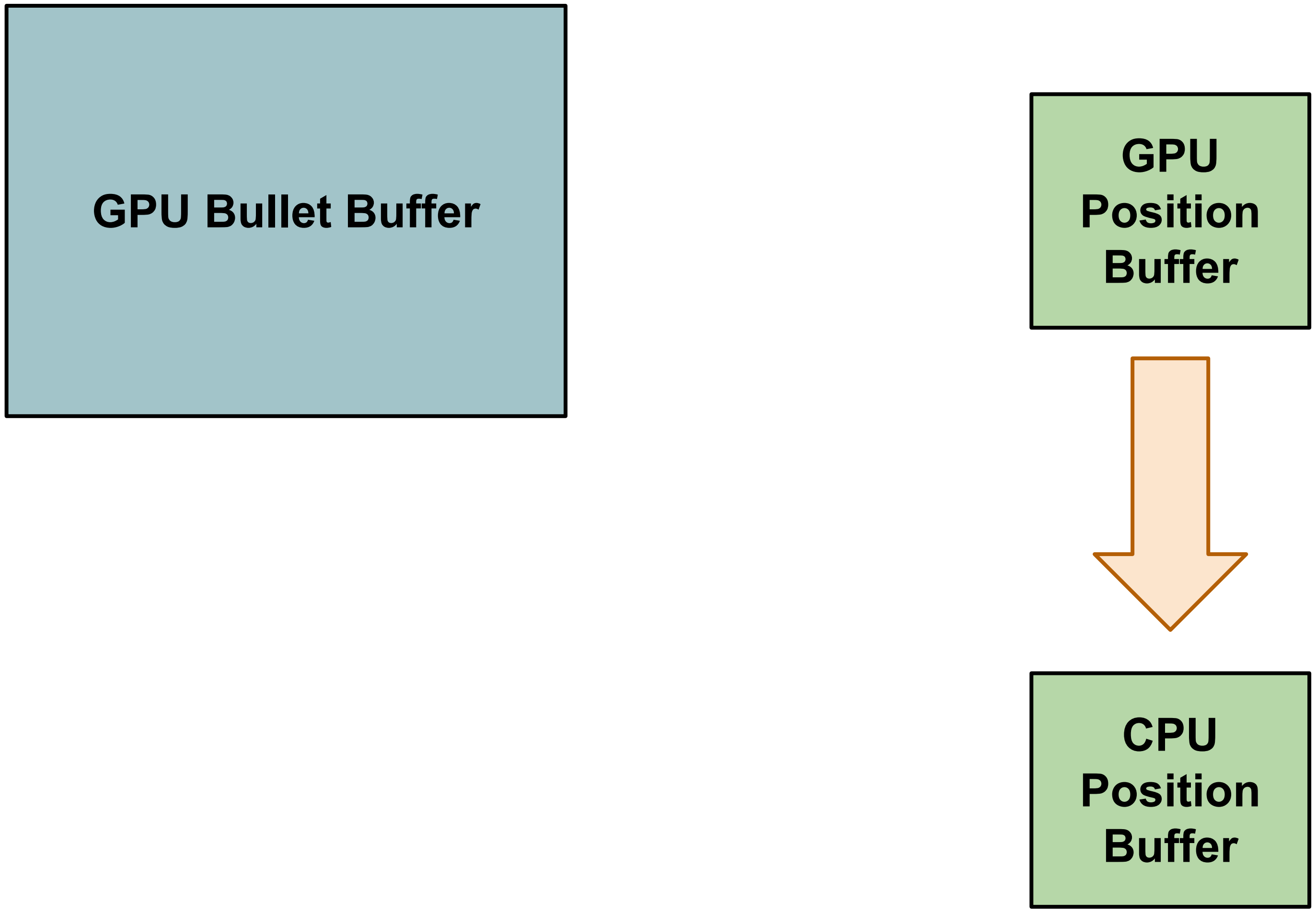
CUDA Concurrency

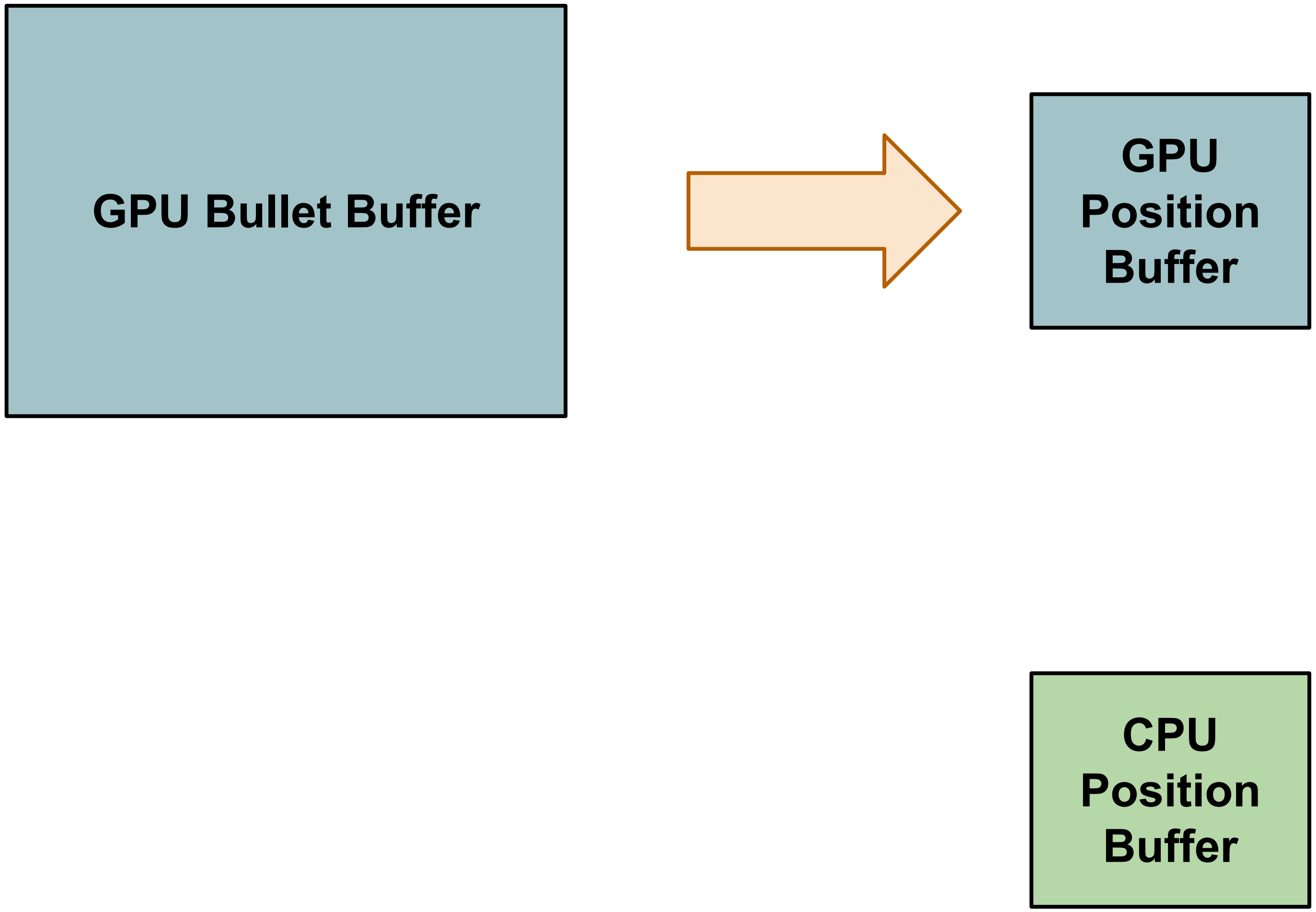
Henry Huang
E190U

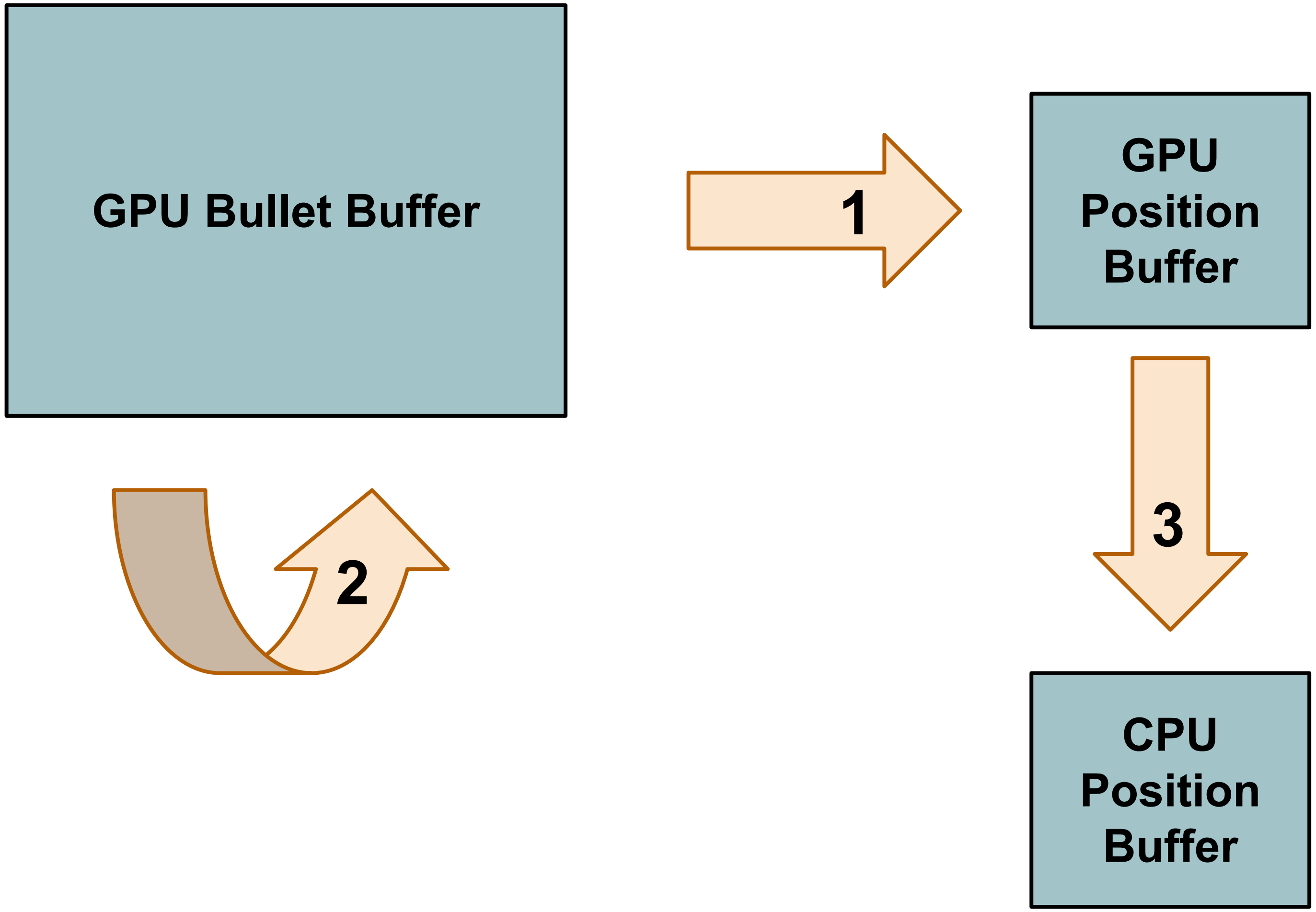
GPU Bullet Buffer


GPU Bullet Buffer
GPU Position Buffer

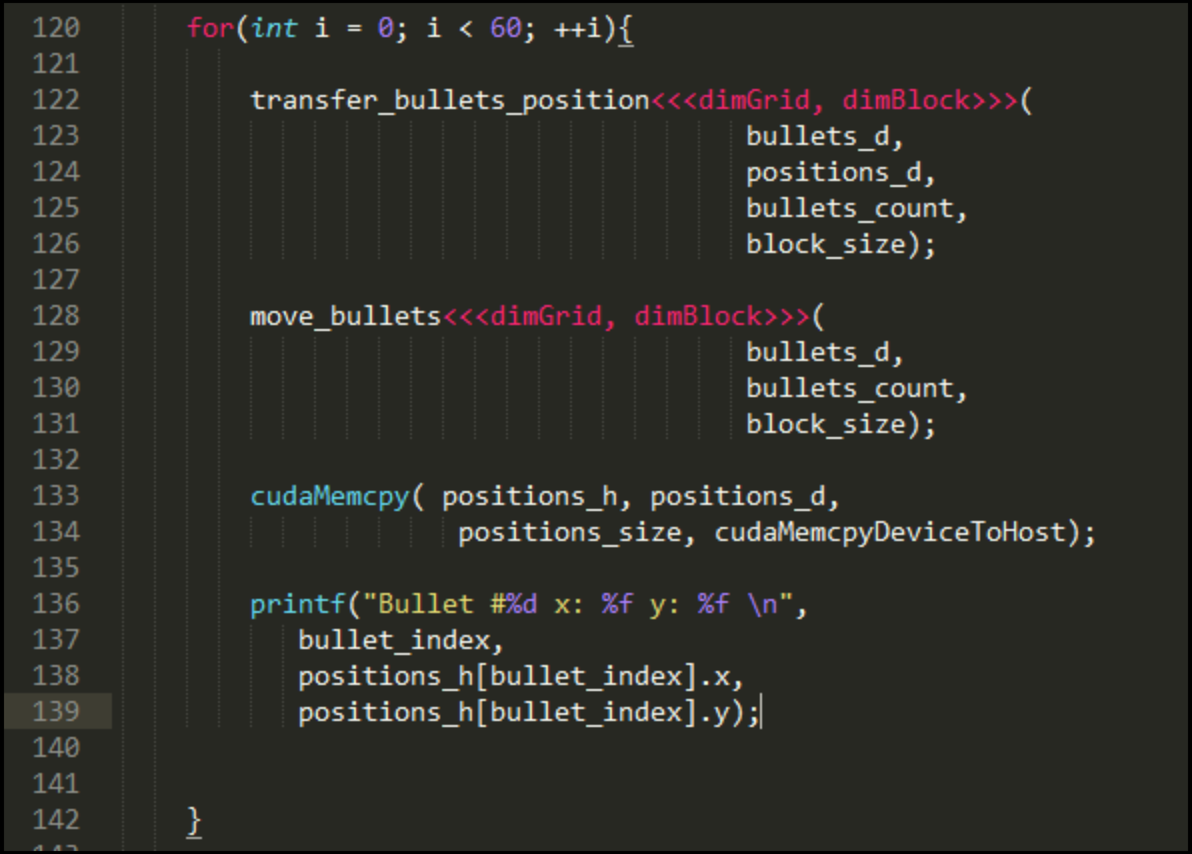

GPU Bullet Buffer
GPU Position Buffer


GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer


GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer


GPU Bullet Buffer
1
GPU Position Buffer
2
3
CPU Position Buffer

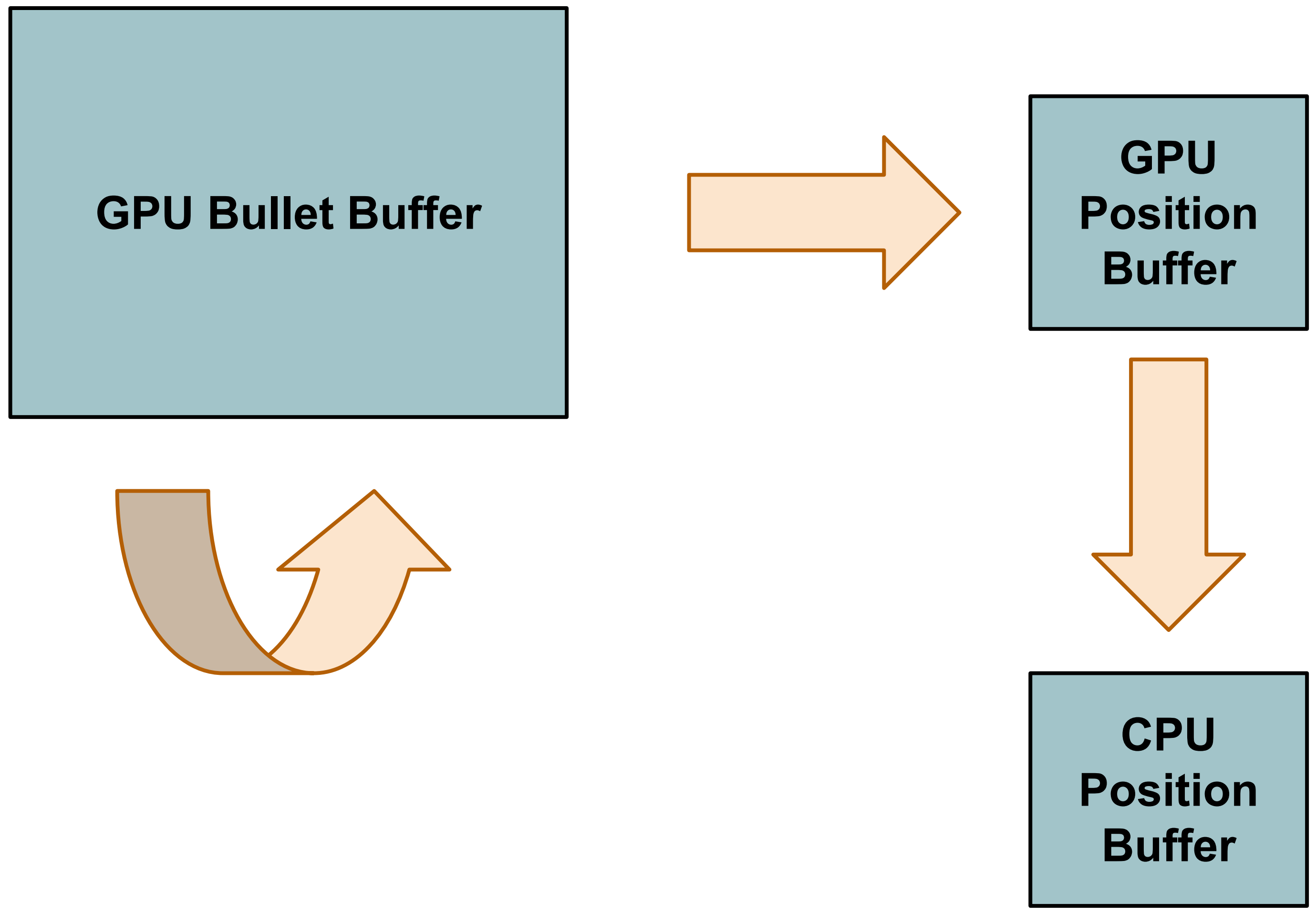
E190U: VIDEO GAME CONSOLE DESIGN



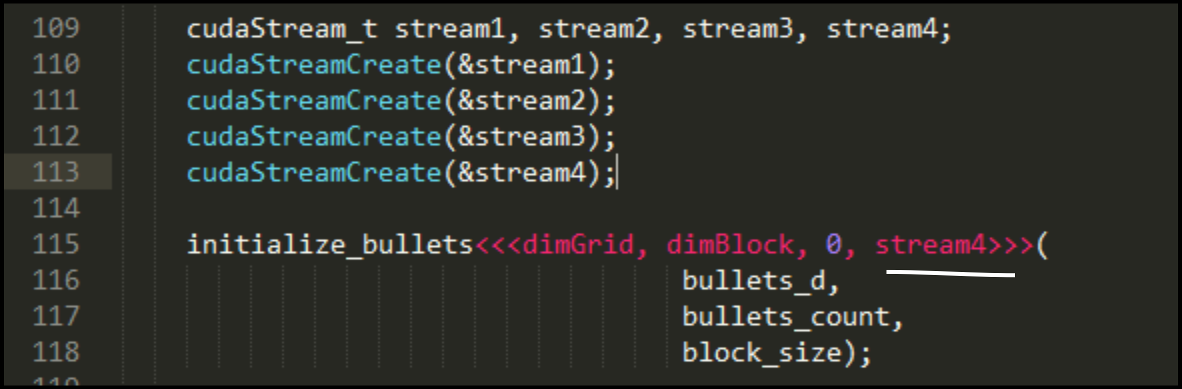
```
for(int i = 0; i < 60; ++i){

    transfer_bullets_position<<<dimGrid, dimBlock>>>(
                                bullets_d,
                                positions_d,
                                bullets_count,
                                block_size);

    move_bullets<<<dimGrid, dimBlock>>>(
                                bullets_d,
                                bullets_count,
                                block_size);

    cudaMemcpy( positions_h, positions_d,
                 positions_size, cudaMemcpyDeviceToHost);

    printf("Bullet #%d x: %f y: %f \n",
       bullet_index,
       positions_h[bullet_index].x,
       positions_h[bullet_index].y);


}
```

Output

```
Bullet #900 x: 900.040100 y: 899.999900
Bullet #900 x: 900.040200 y: 899.999800
Bullet #900 x: 900.040300 y: 899.999700
Bullet #900 x: 900.040400 y: 899.999600
Bullet #900 x: 900.040500 y: 899.999500
Bullet #900 x: 900.040600 y: 899.999400
Bullet #900 x: 900.040700 y: 899.999300
Bullet #900 x: 900.040800 y: 899.999200
Bullet #900 x: 900.040900 y: 899.999100
Bullet #900 x: 900.041000 y: 899.999000
Bullet #900 x: 900.041100 y: 899.998900
```

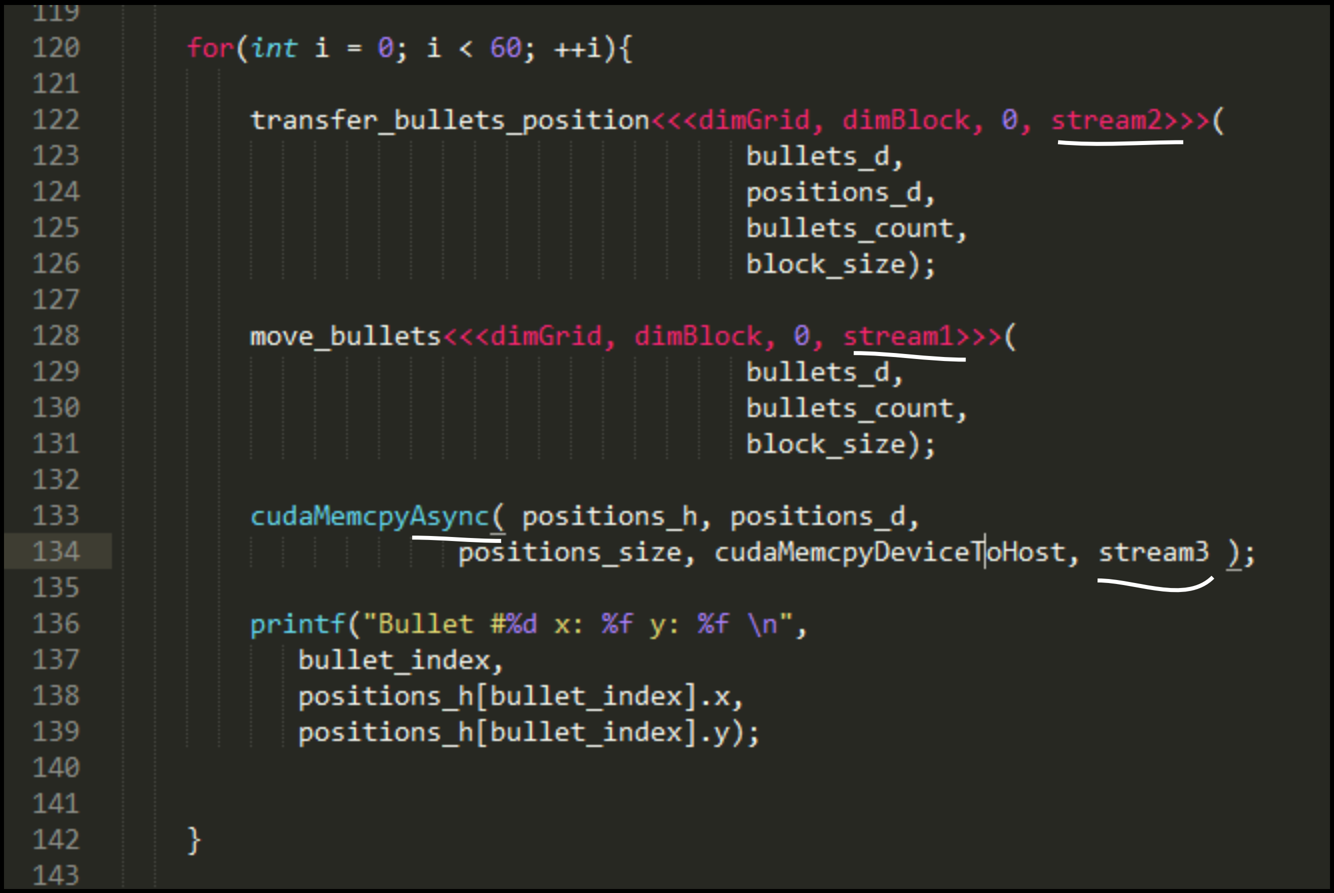

GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer

E190U: VIDEO GAME CONSOLE DESIGN

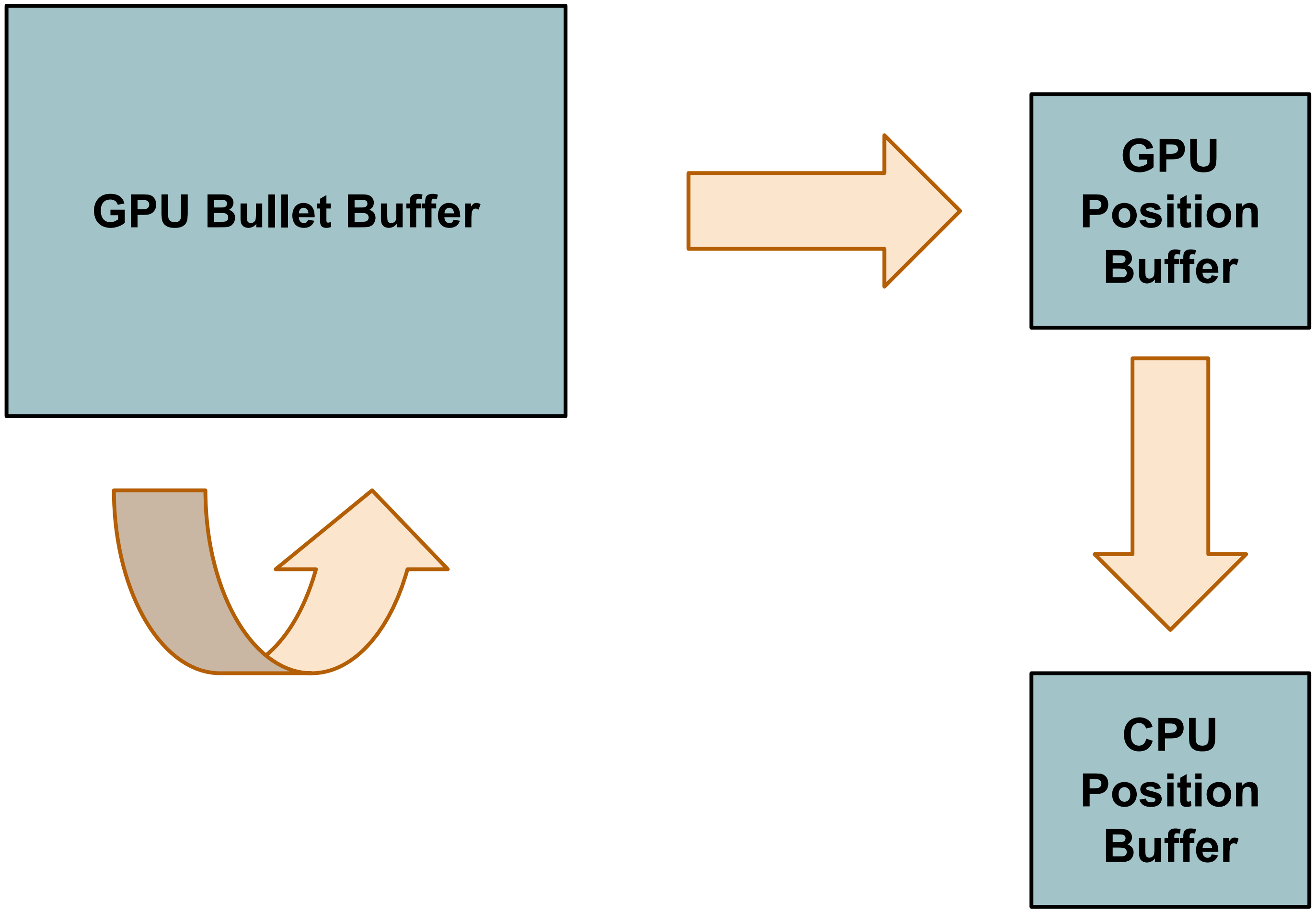


```
    cudaStream_t stream1, stream2, stream3, stream4;
    cudaStreamCreate(&stream1);
    cudaStreamCreate(&stream2);
    cudaStreamCreate(&stream3);
    cudaStreamCreate(&stream4);

    initialize_bullets<<<dimGrid, dimBlock, 0, stream4>>>(
                                                bullets_d,
                                                bullets_count,
                                                block_size);
```

E190U: VIDEO GAME CONSOLE DESIGN

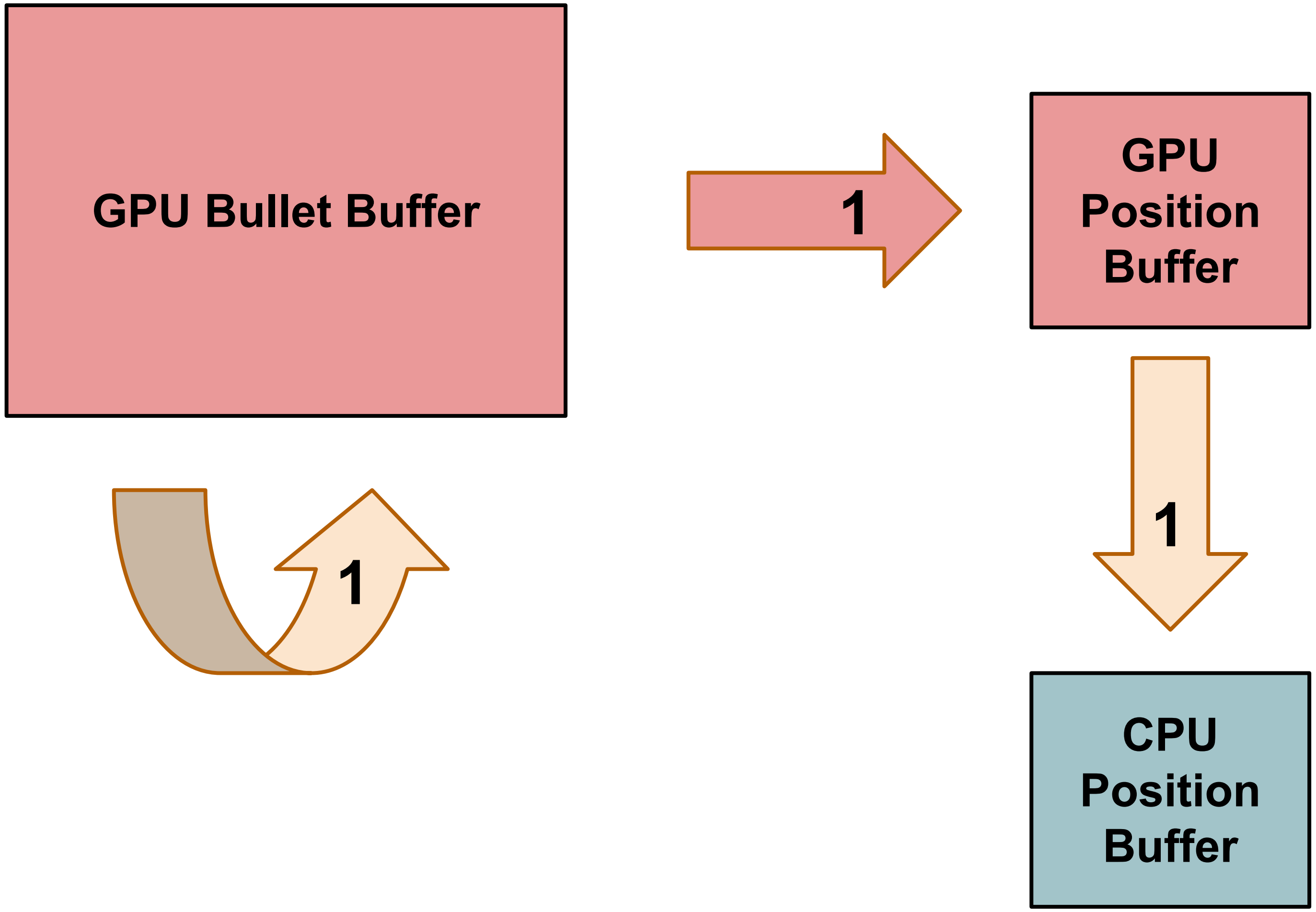


```
for(int i = 0; i < 60; ++i){

    transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(
                                bullets_d,
                                positions_d,
                                bullets_count,
                                block_size);

    move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(
                                bullets_d,
                                bullets_count,
                                block_size);

    cudaMemcpyAsync( positions_h, positions_d,
                positions_size, cudaMemcpyDeviceToHost, stream3 );

    printf("Bullet #%d x: %f y: %f \n",
        bullet_index,
        positions_h[bullet_index].x,
        positions_h[bullet_index].y);


}
```

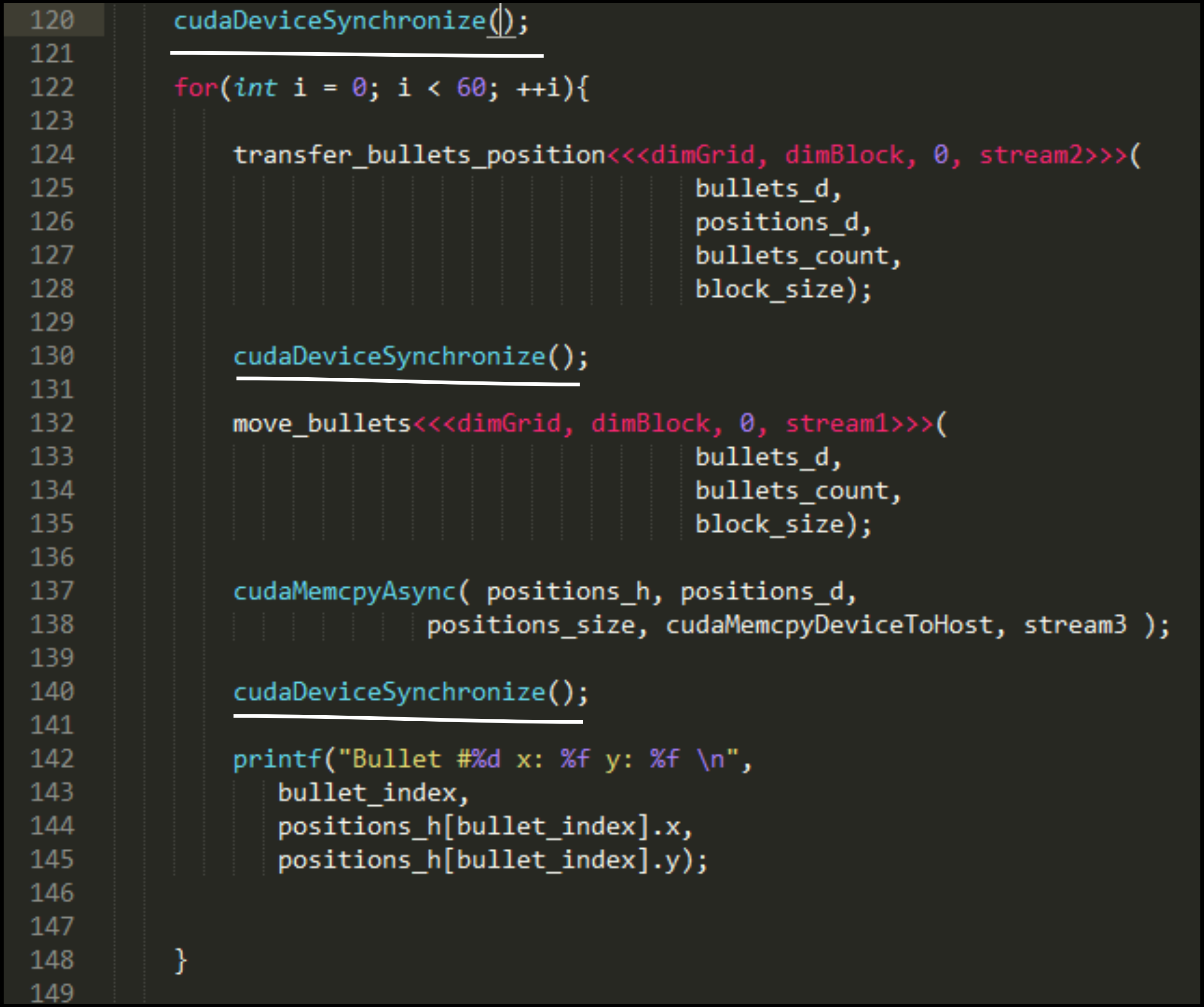

GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer

Output

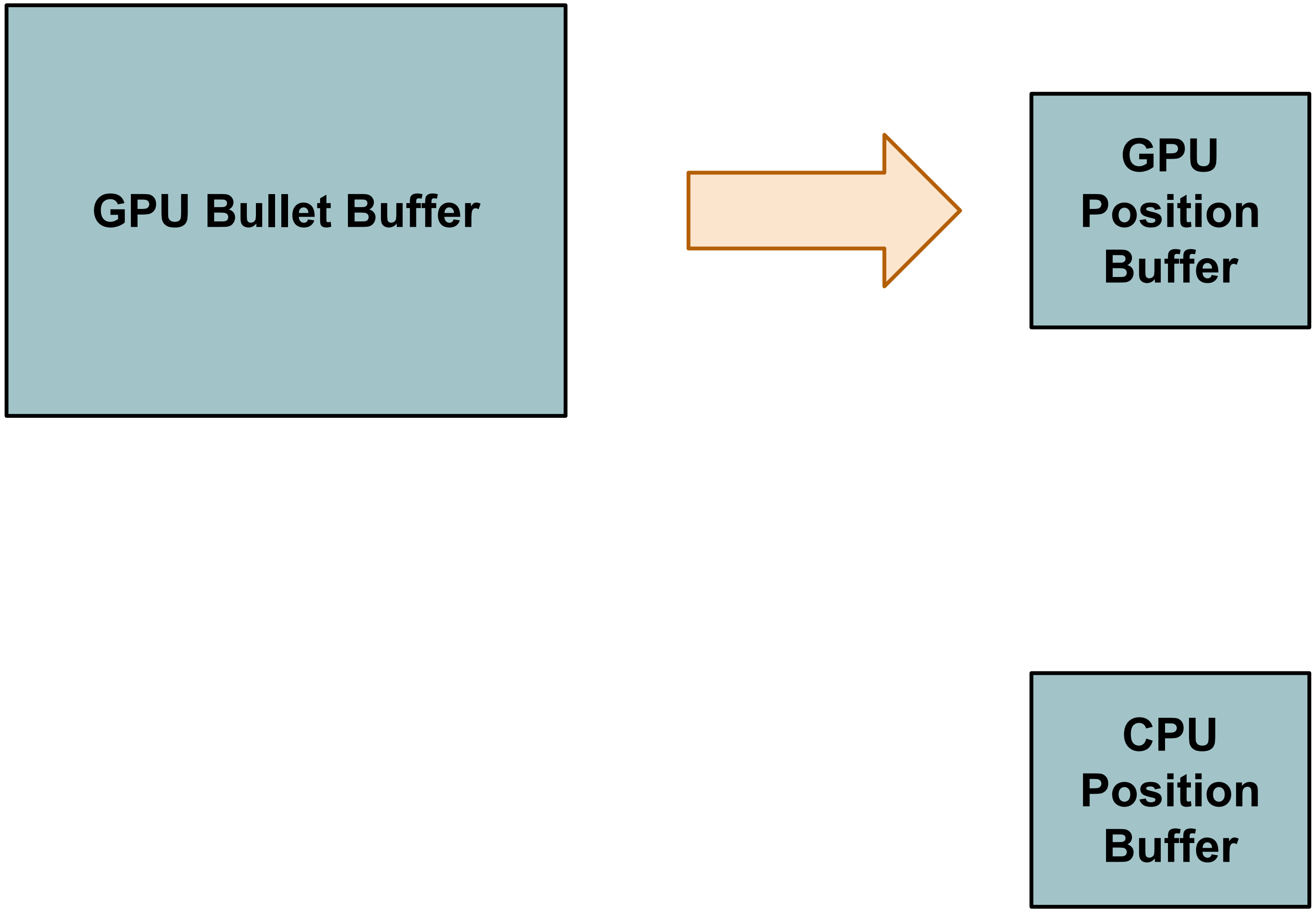
```
Bullet #900 x: 900.040100 y: 899.999900
Bullet #900 x: 900.040200 y: 899.999800
Bullet #900 x: 900.040300 y: 899.999700
Bullet #900 x: 900.040400 y: 899.999600
Bullet #900 x: 900.040500 y: 899.999500
Bullet #900 x: 900.040600 y: 899.999400
Bullet #900 x: 900.040600 y: 899.999400
Bullet #900 x: 900.040800 y: 899.999200
Bullet #900 x: 900.040900 y: 899.999100
Bullet #900 x: 900.041000 y: 899.999000
Bullet #900 x: 900.041000 y: 899.999000
```

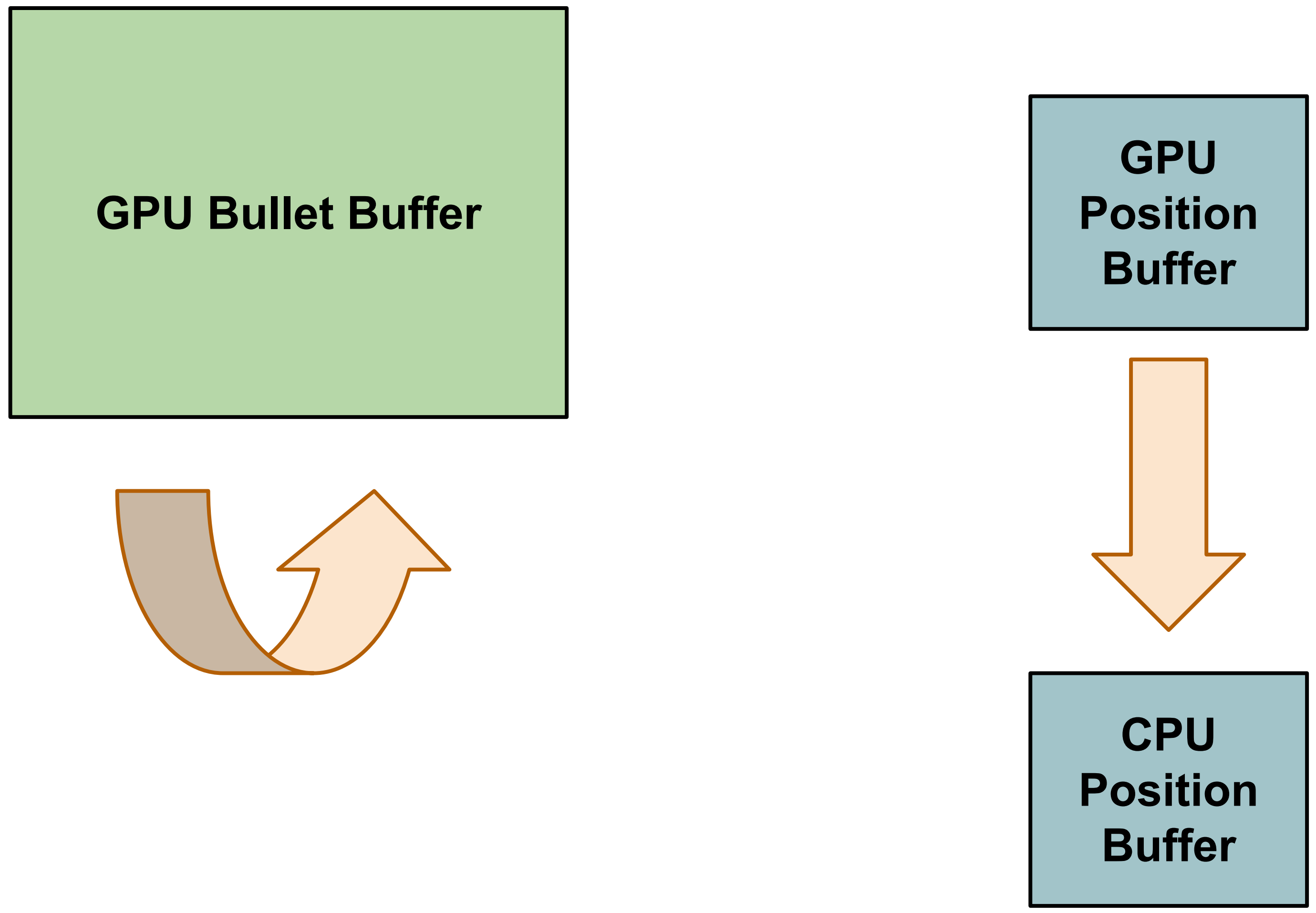

GPU Bullet Buffer
1
1
GPU Position Buffer
1
CPU Position Buffer

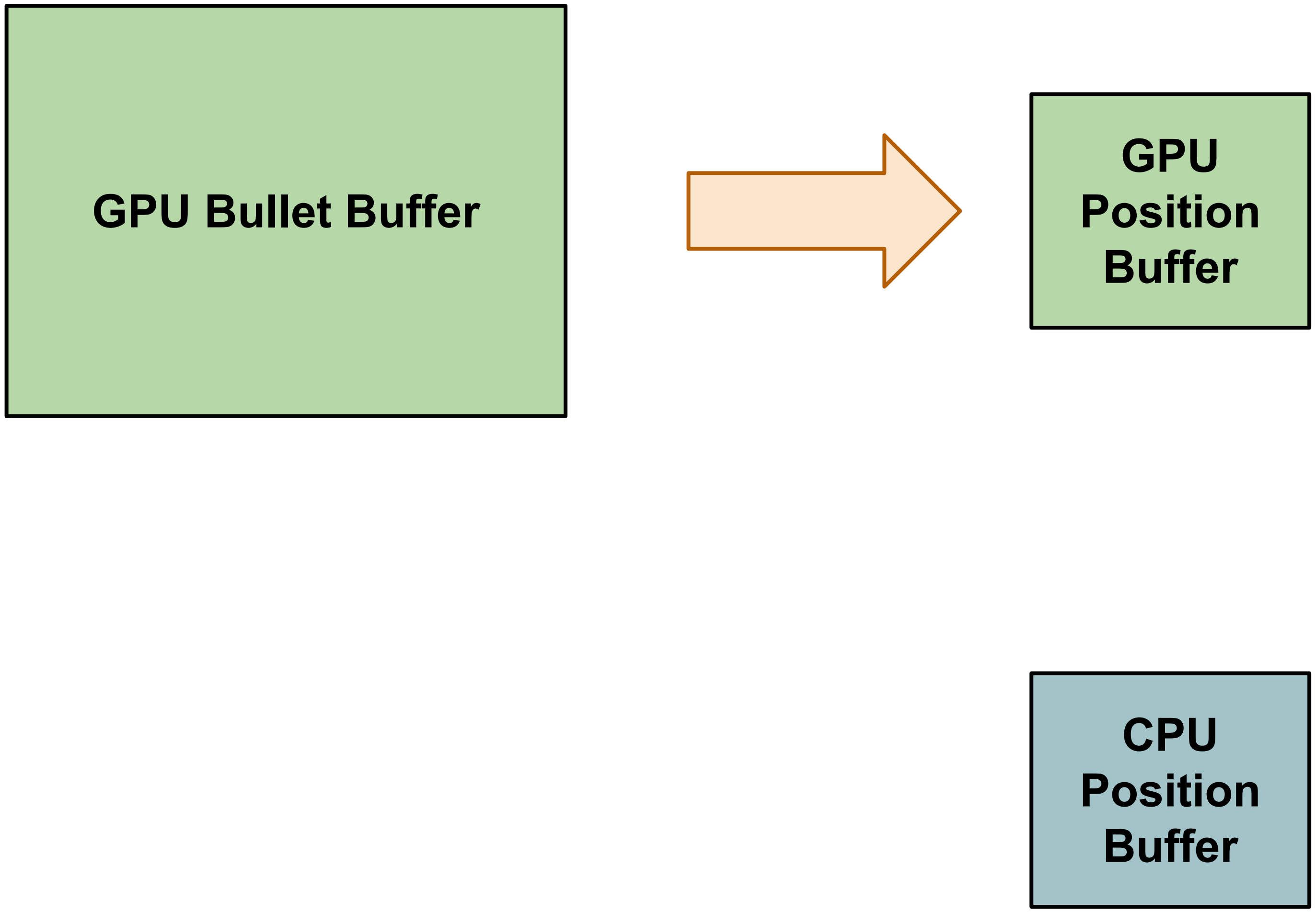
E190U: VIDEO GAME CONSOLE DESIGN

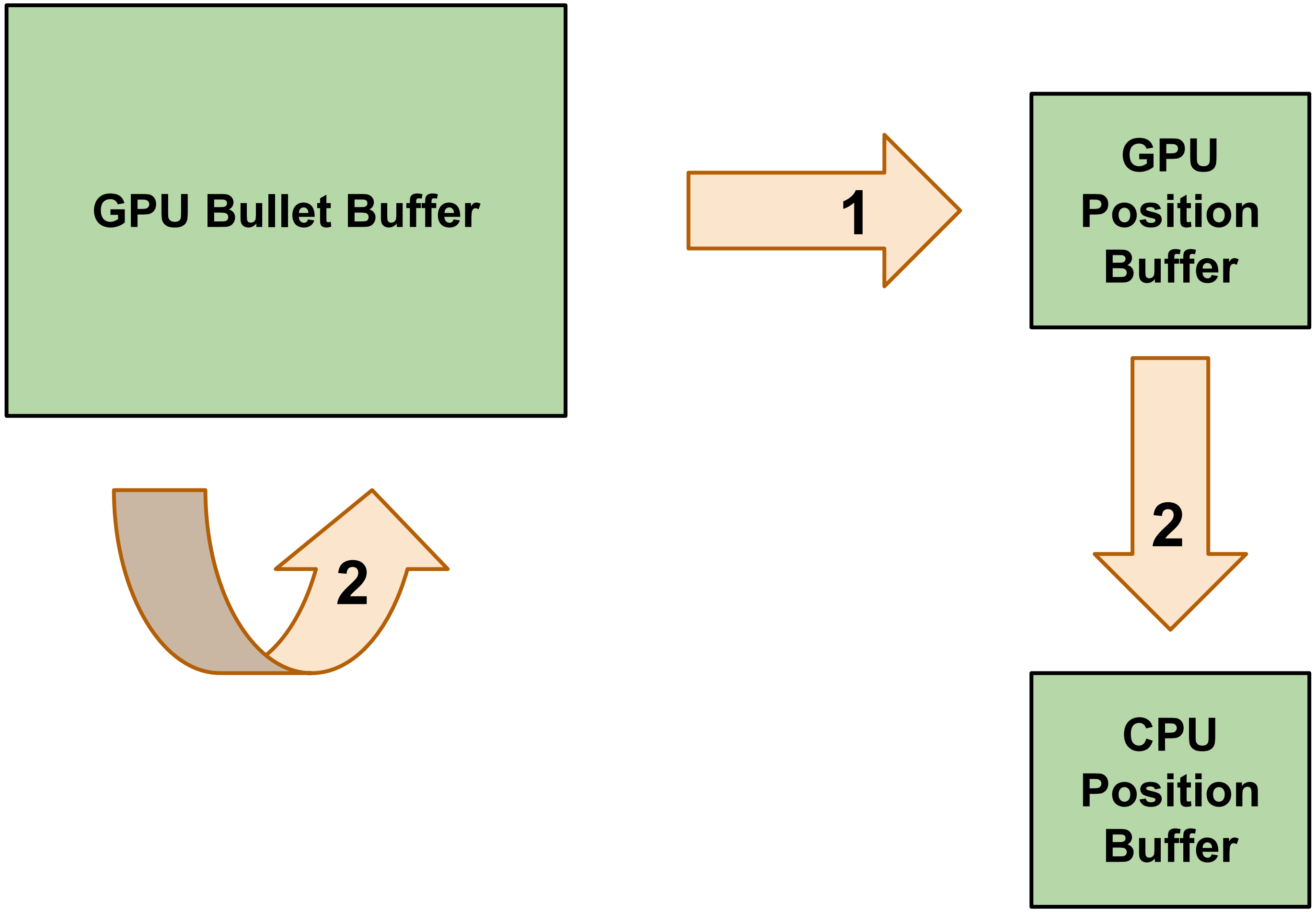


```
    cudaDeviceSynchronize();

    for(int i = 0; i < 60; ++i){

        transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(
                                                bullets_d,
                                                positions_d,
                                                bullets_count,
                                                block_size);

        cudaDeviceSynchronize();

        move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(
                                                bullets_d,
                                                bullets_count,
                                                block_size);

        cudaMemcpyAsync( positions_h, positions_d,
                      positions_size, cudaMemcpyDeviceToHost, stream3 );

        cudaDeviceSynchronize();

        printf("Bullet #%d x: %f y: %f \n",
           bullet_index,
           positions_h[bullet_index].x,
           positions_h[bullet_index].y);


    }

```

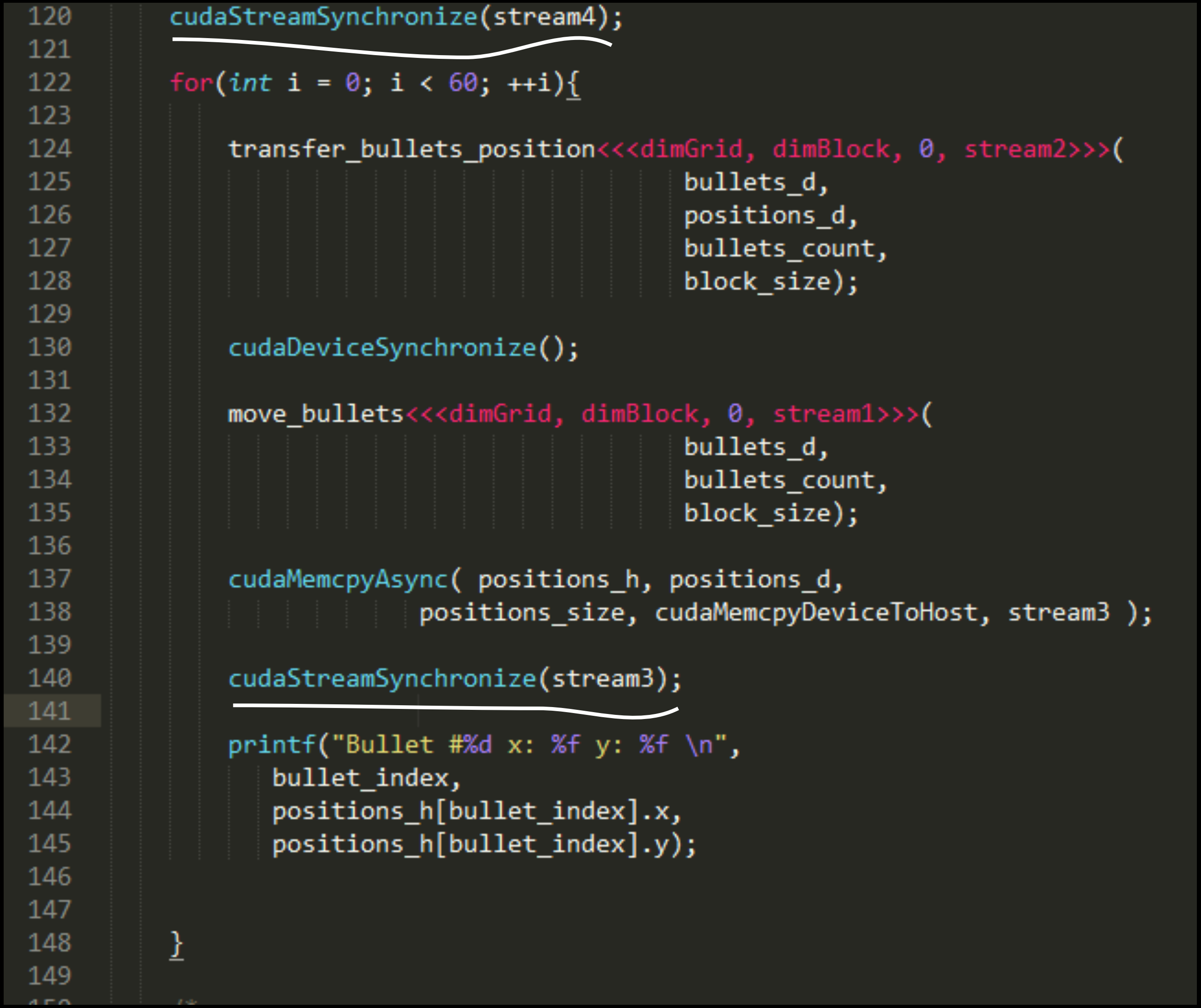

GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer


GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer


GPU Bullet Buffer
GPU Position Buffer
CPU Position Buffer


GPU Bullet Buffer
1
GPU Position Buffer
2
CPU Position Buffer
2

E190U: VIDEO GAME CONSOLE DESIGN



```
    cudaStreamSynchronize(stream4);

    for(int i = 0; i < 60; ++i){

        transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(
                                   bullets_d,
                                   positions_d,
                                   bullets_count,
                                   block_size);

        cudaDeviceSynchronize();

        move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(
                                   bullets_d,
                                   bullets_count,
                                   block_size);

        cudaMemcpyAsync( positions_h, positions_d,
              positions_size, cudaMemcpyDeviceToHost, stream3 );

        cudaStreamSynchronize(stream3);

        printf("Bullet #%d x: %f y: %f \n",
          bullet_index,
          positions_h[bullet_index].x,
          positions_h[bullet_index].y);


    }
```

Summary

- Each cudaStream_t allows a different control flow
- cudaMemcpyAsync() works during computation
- Synchronization must be manually forced
- cudaStreamSynchronize() waits for stream to end
- cudaDeviceSynchronize() waits for all streams to end

More Information

Cuda C/C++ Streams and Concurrency