

CUDA Concurrency

Henry Huang

E190U

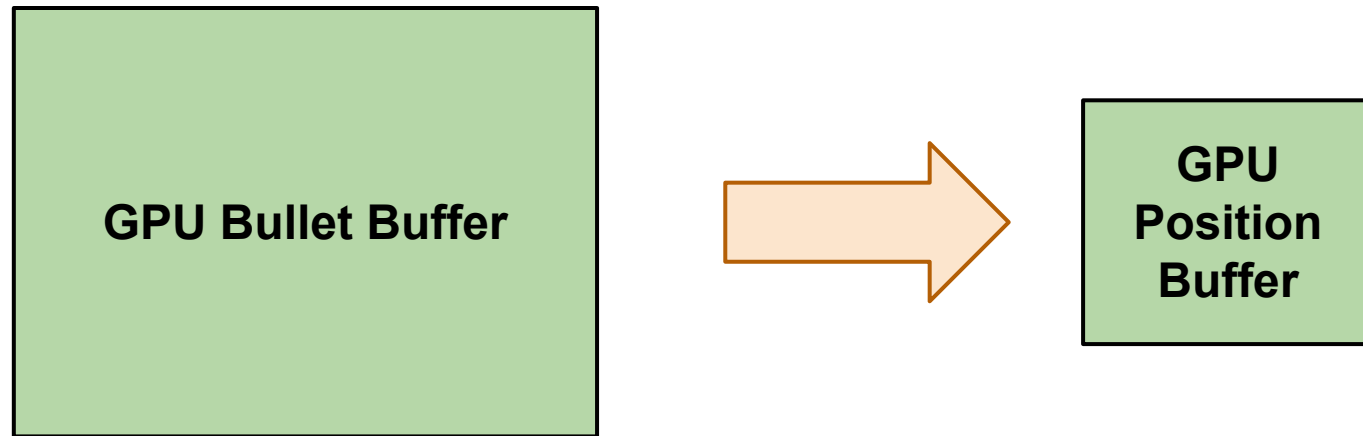
E190U: VIDEO GAME CONSOLE DESIGN



A diagram consisting of a light green square with a black border. The text "GPU Bullet Buffer" is centered within the square.

GPU Bullet Buffer

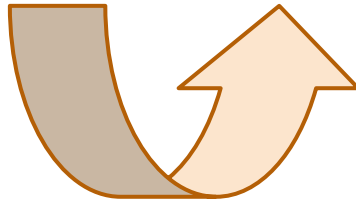
E190U: VIDEO GAME CONSOLE DESIGN



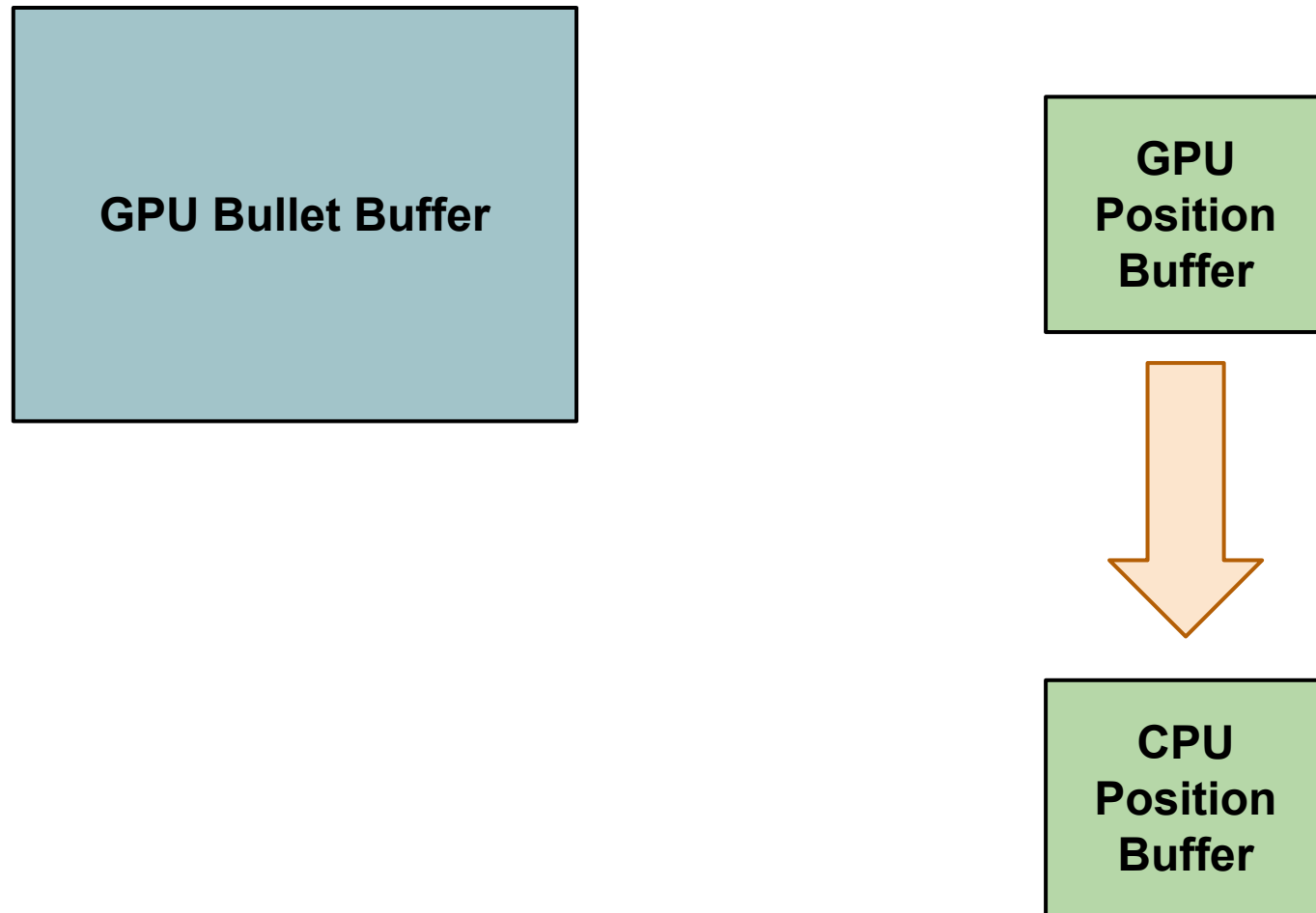
E190U: VIDEO GAME CONSOLE DESIGN

GPU Bullet Buffer

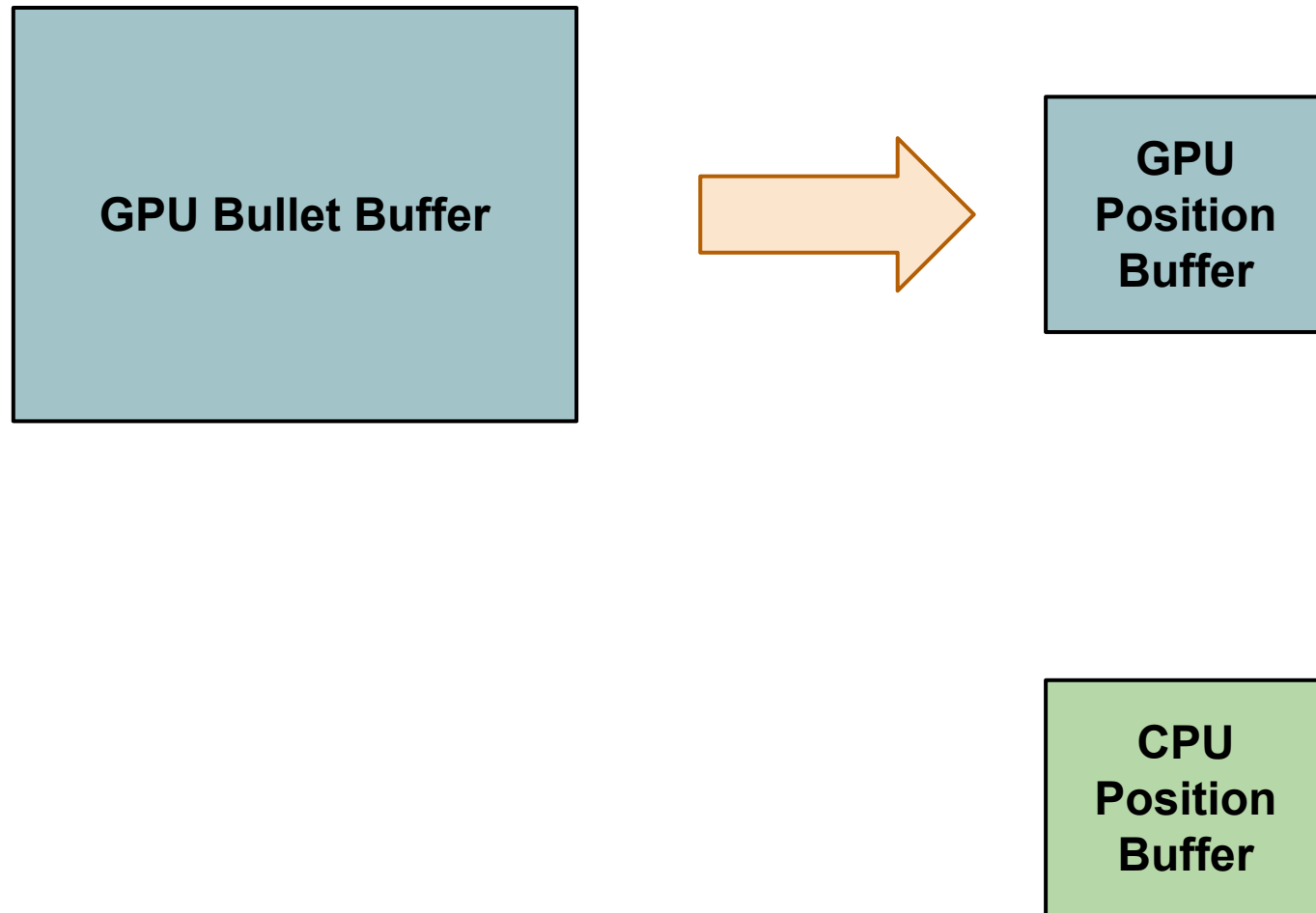
**GPU
Position
Buffer**



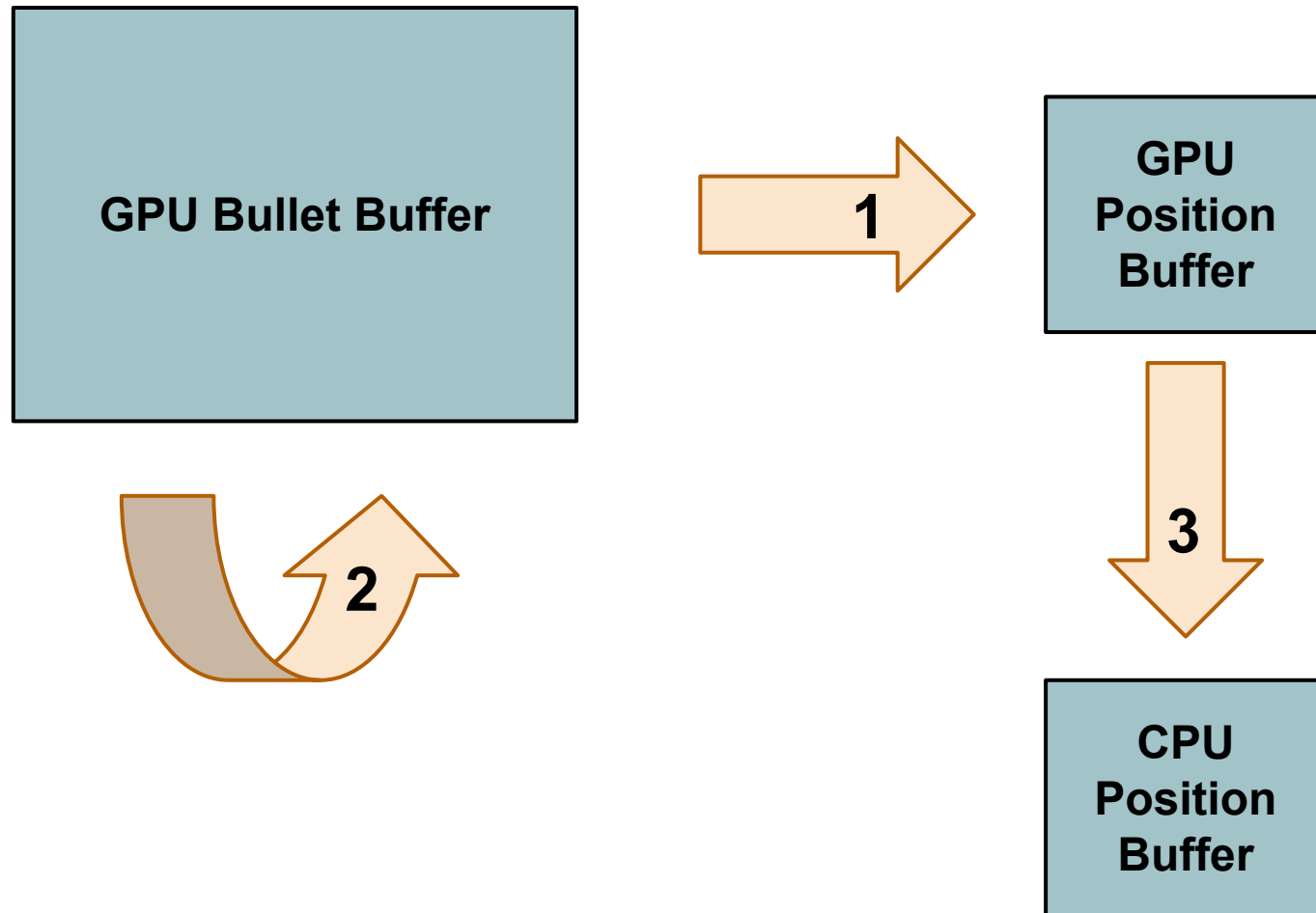
E190U: VIDEO GAME CONSOLE DESIGN



E190U: VIDEO GAME CONSOLE DESIGN



E190U: VIDEO GAME CONSOLE DESIGN



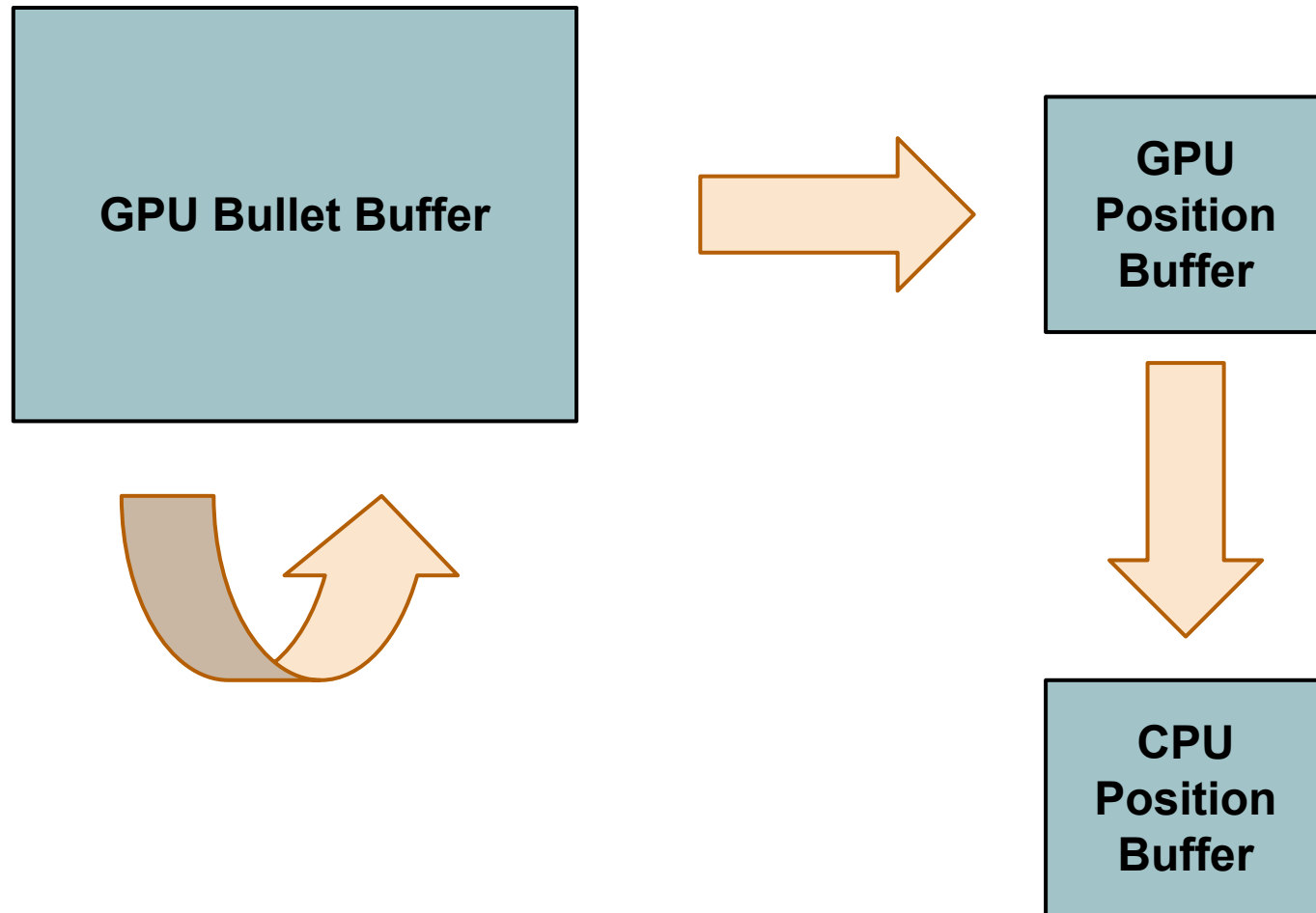
E190U: VIDEO GAME CONSOLE DESIGN

```
120     for(int i = 0; i < 60; ++i){
121
122         transfer_bullets_position<<<dimGrid, dimBlock>>>(
123             bullets_d,
124             positions_d,
125             bullets_count,
126             block_size);
127
128         move_bullets<<<dimGrid, dimBlock>>>(
129             bullets_d,
130             bullets_count,
131             block_size);
132
133         cudaMemcpy( positions_h, positions_d,
134             positions_size, cudaMemcpyDeviceToHost);
135
136         printf("Bullet #%d x: %f y: %f \n",
137             bullet_index,
138             positions_h[bullet_index].x,
139             positions_h[bullet_index].y);
140
141
142     }
```


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

E190U: VIDEO GAME CONSOLE DESIGN



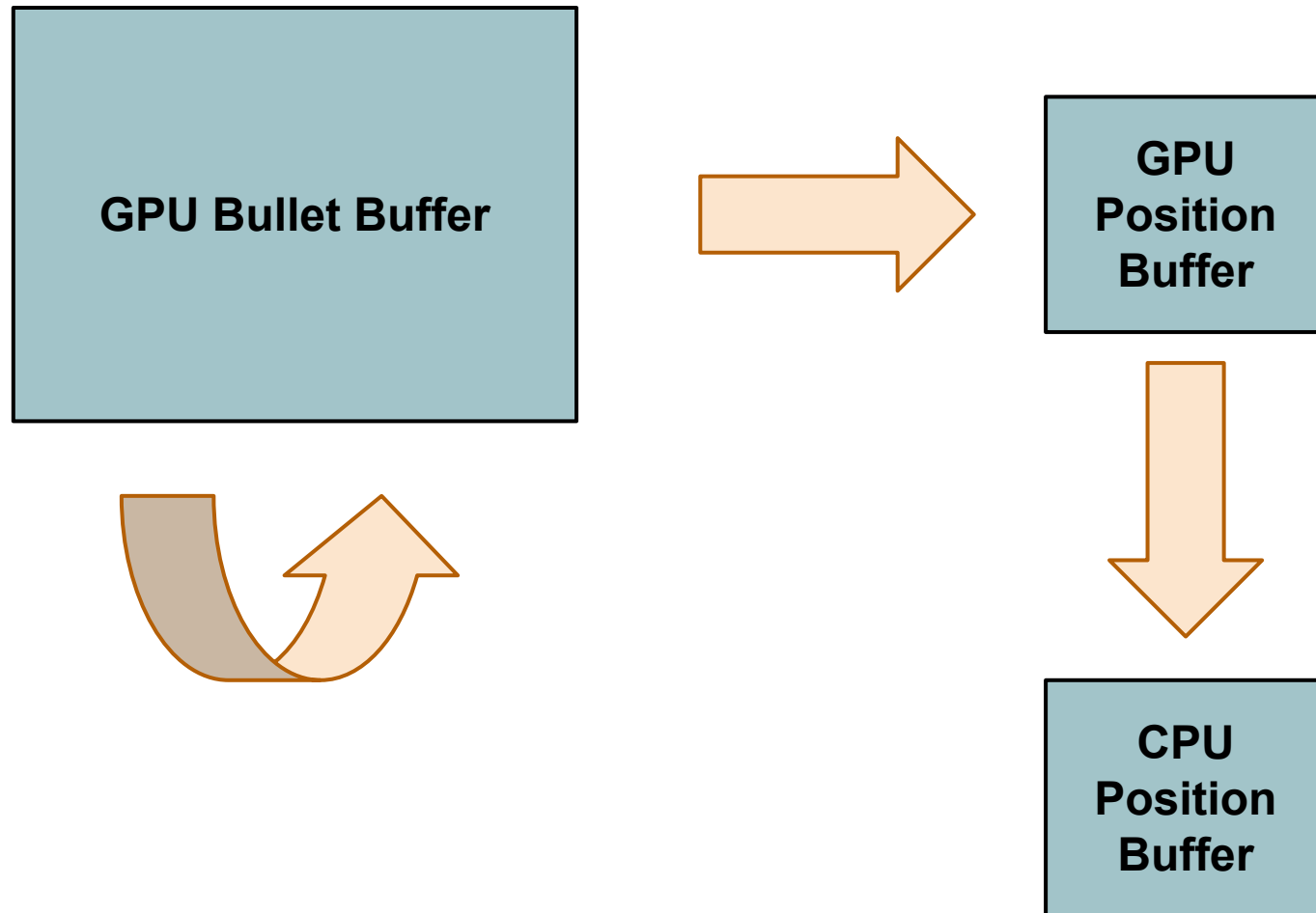
E190U: VIDEO GAME CONSOLE DESIGN

```
109     cudaStream_t stream1, stream2, stream3, stream4;
110     cudaStreamCreate(&stream1);
111     cudaStreamCreate(&stream2);
112     cudaStreamCreate(&stream3);
113     cudaStreamCreate(&stream4);
114
115     initialize_bullets<<<dimGrid, dimBlock, 0, stream4>>>(  
116         bullets_d,                       
117         bullets_count,  
118         block_size);  
119
```

E190U: VIDEO GAME CONSOLE DESIGN

```
119
120     for(int i = 0; i < 60; ++i){
121
122         transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(  
123             bullets_d,  
124             positions_d,  
125             bullets_count,  
126             block_size);
127
128         move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(  
129             bullets_d,  
130             bullets_count,  
131             block_size);
132
133         cudaMemcpyAsync( positions_h, positions_d,  
134             positions_size, cudaMemcpyDeviceToHost, stream3 );
135
136         printf("Bullet #%d x: %f y: %f \n",  
137             bullet_index,  
138             positions_h[bullet_index].x,  
139             positions_h[bullet_index].y);
140
141
142     }
143
```

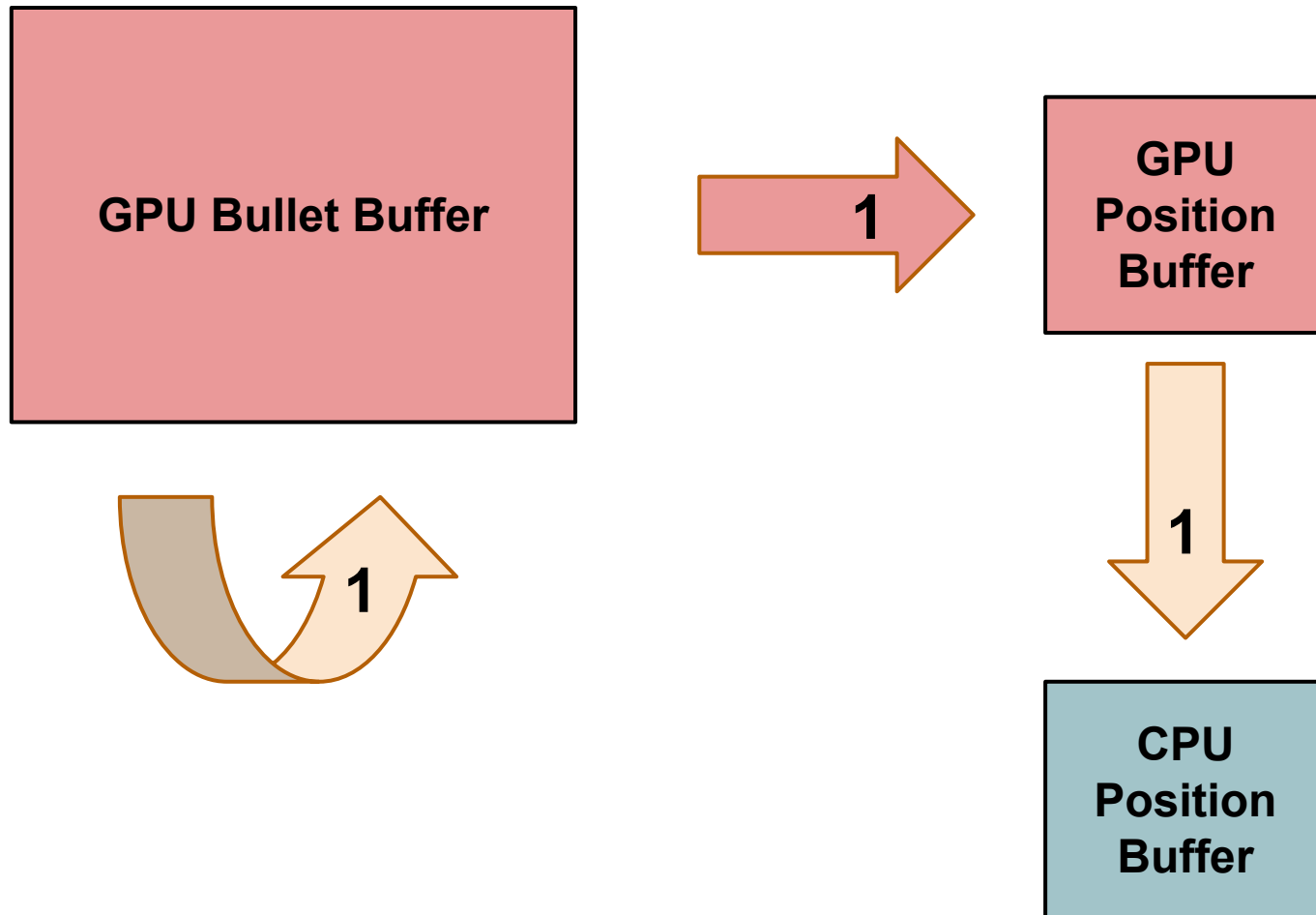
E190U: VIDEO GAME CONSOLE DESIGN



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

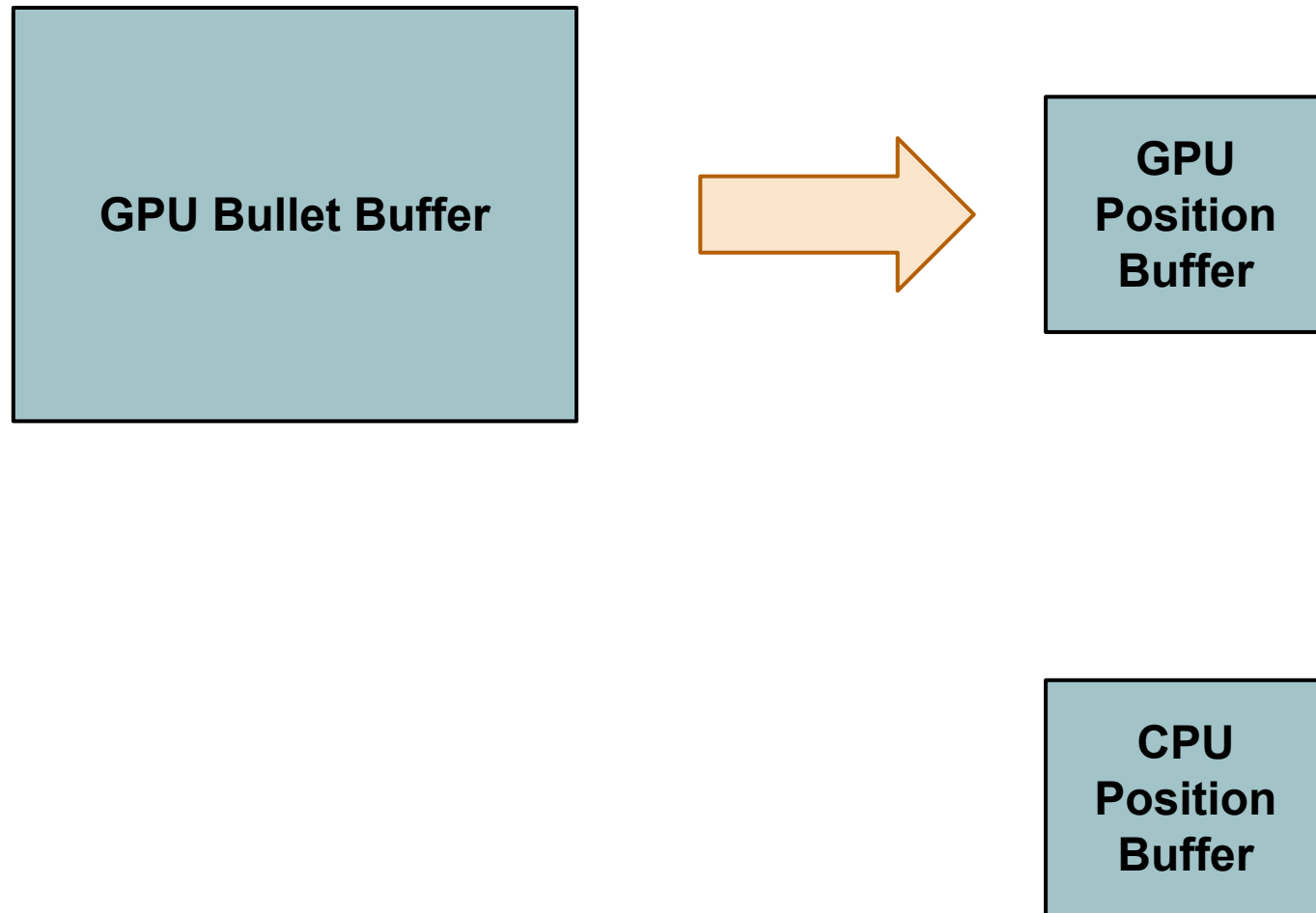
E190U: VIDEO GAME CONSOLE DESIGN



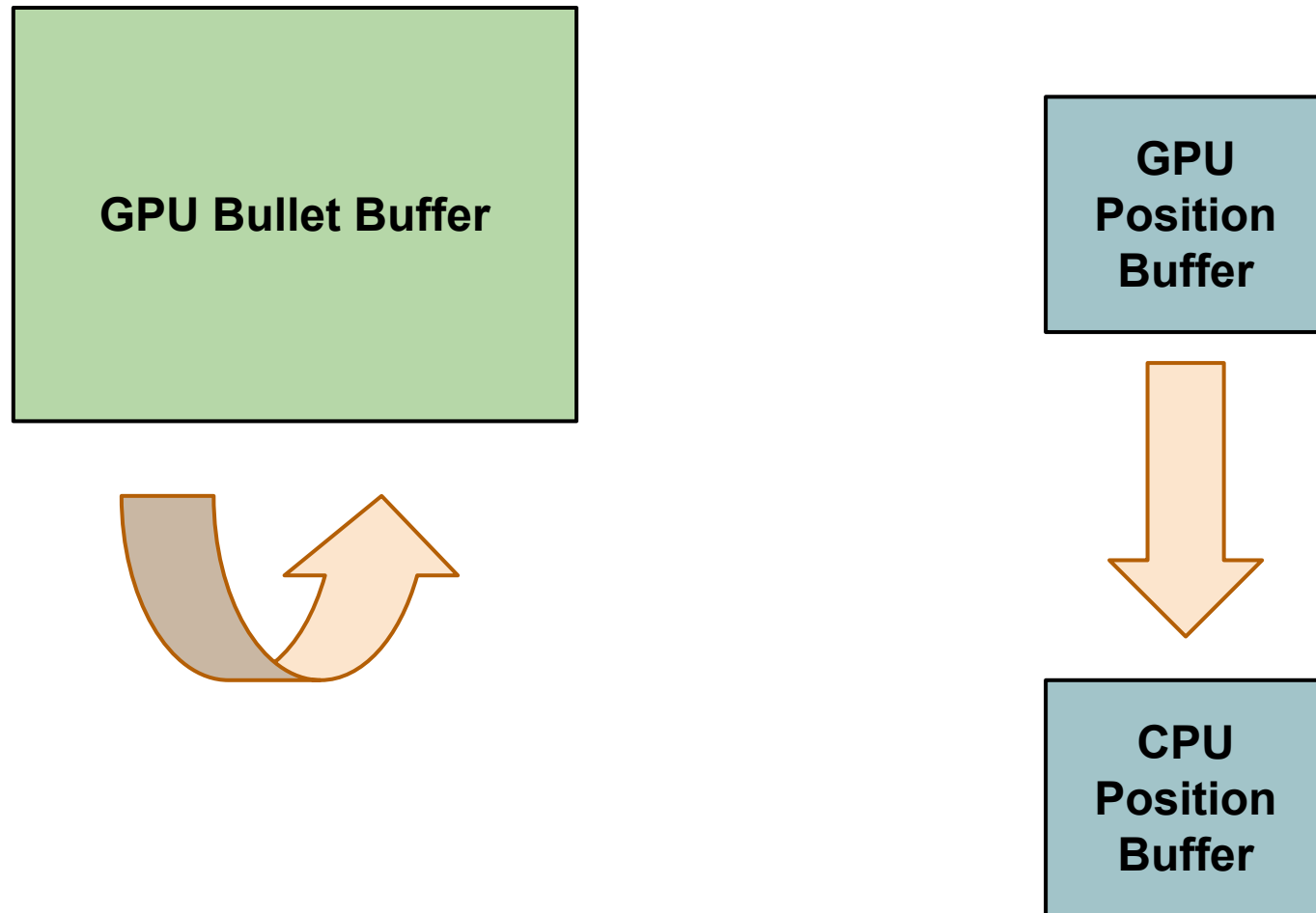
E190U: VIDEO GAME CONSOLE DESIGN

```
120     cudaDeviceSynchronize();
121
122     for(int i = 0; i < 60; ++i){
123
124         transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(
125             bullets_d,
126             positions_d,
127             bullets_count,
128             block_size);
129
130         cudaDeviceSynchronize();
131
132         move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(
133             bullets_d,
134             bullets_count,
135             block_size);
136
137         cudaMemcpyAsync( positions_h, positions_d,
138             positions_size, cudaMemcpyDeviceToHost, stream3 );
139
140         cudaDeviceSynchronize();
141
142         printf("Bullet #%d x: %f y: %f \n",
143             bullet_index,
144             positions_h[bullet_index].x,
145             positions_h[bullet_index].y);
146
147
148     }
149
```

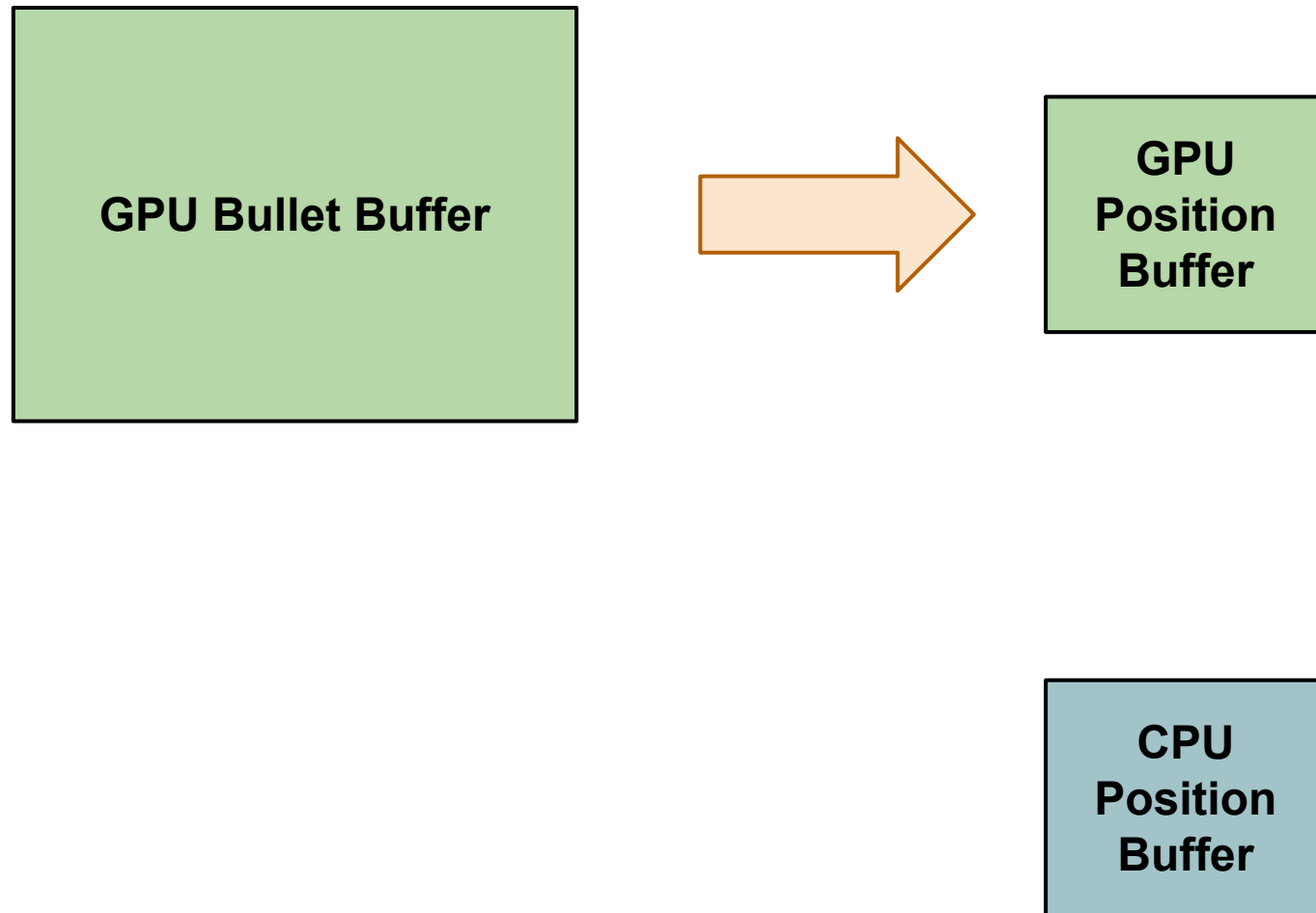

E190U: VIDEO GAME CONSOLE DESIGN



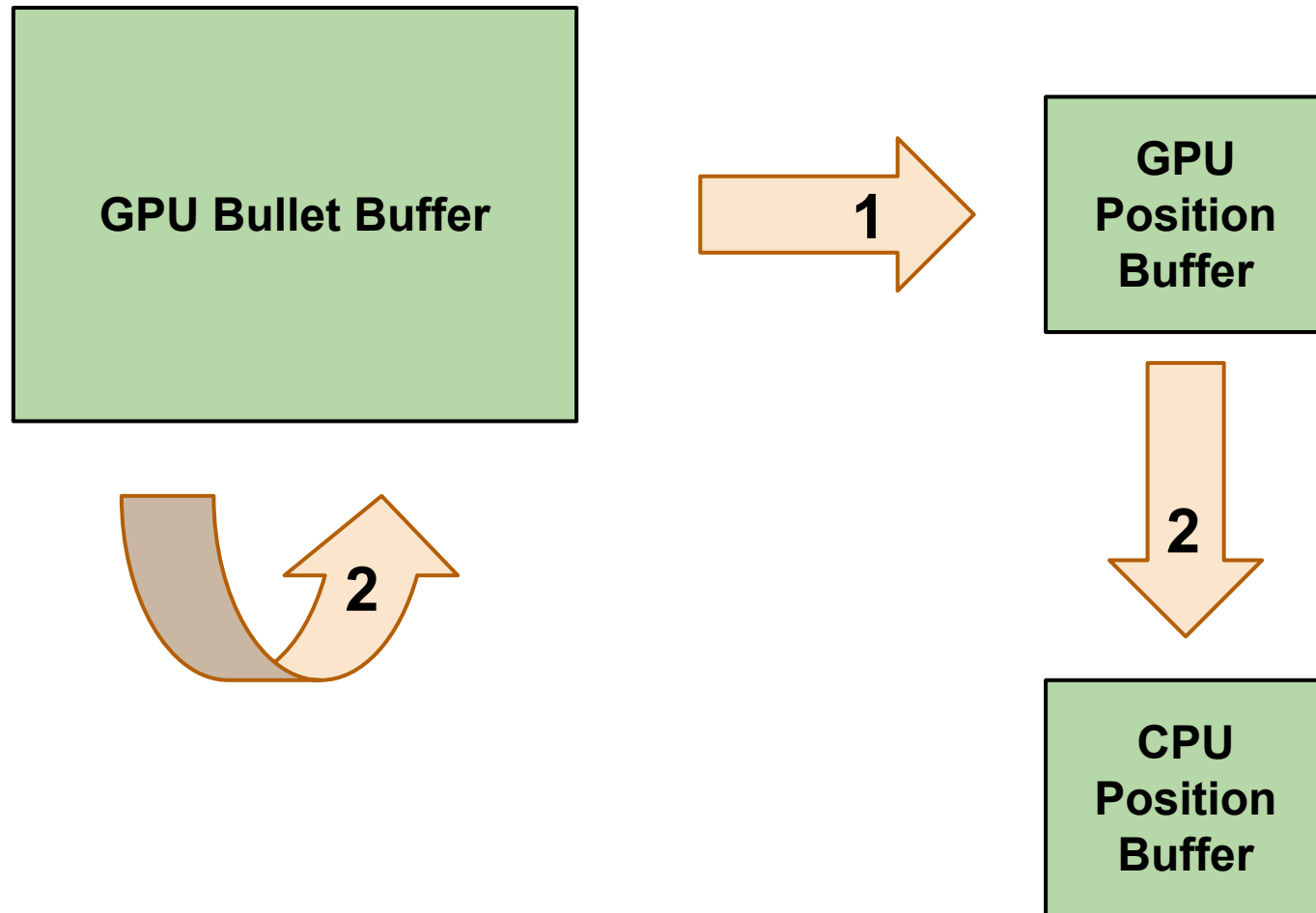
E190U: VIDEO GAME CONSOLE DESIGN



E190U: VIDEO GAME CONSOLE DESIGN



E190U: VIDEO GAME CONSOLE DESIGN



E190U: VIDEO GAME CONSOLE DESIGN

```
120     cudaStreamSynchronize(stream4);
121
122     for(int i = 0; i < 60; ++i){
123
124         transfer_bullets_position<<<dimGrid, dimBlock, 0, stream2>>>(
125             bullets_d,
126             positions_d,
127             bullets_count,
128             block_size);
129
130         cudaDeviceSynchronize();
131
132         move_bullets<<<dimGrid, dimBlock, 0, stream1>>>(
133             bullets_d,
134             bullets_count,
135             block_size);
136
137         cudaMemcpyAsync( positions_h, positions_d,
138             positions_size, cudaMemcpyDeviceToHost, stream3 );
139
140         cudaStreamSynchronize(stream3);
141
142         printf("Bullet #%d x: %f y: %f \n",
143             bullet_index,
144             positions_h[bullet_index].x,
145             positions_h[bullet_index].y);
146
147
148     }
149
150 }
```

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](#)