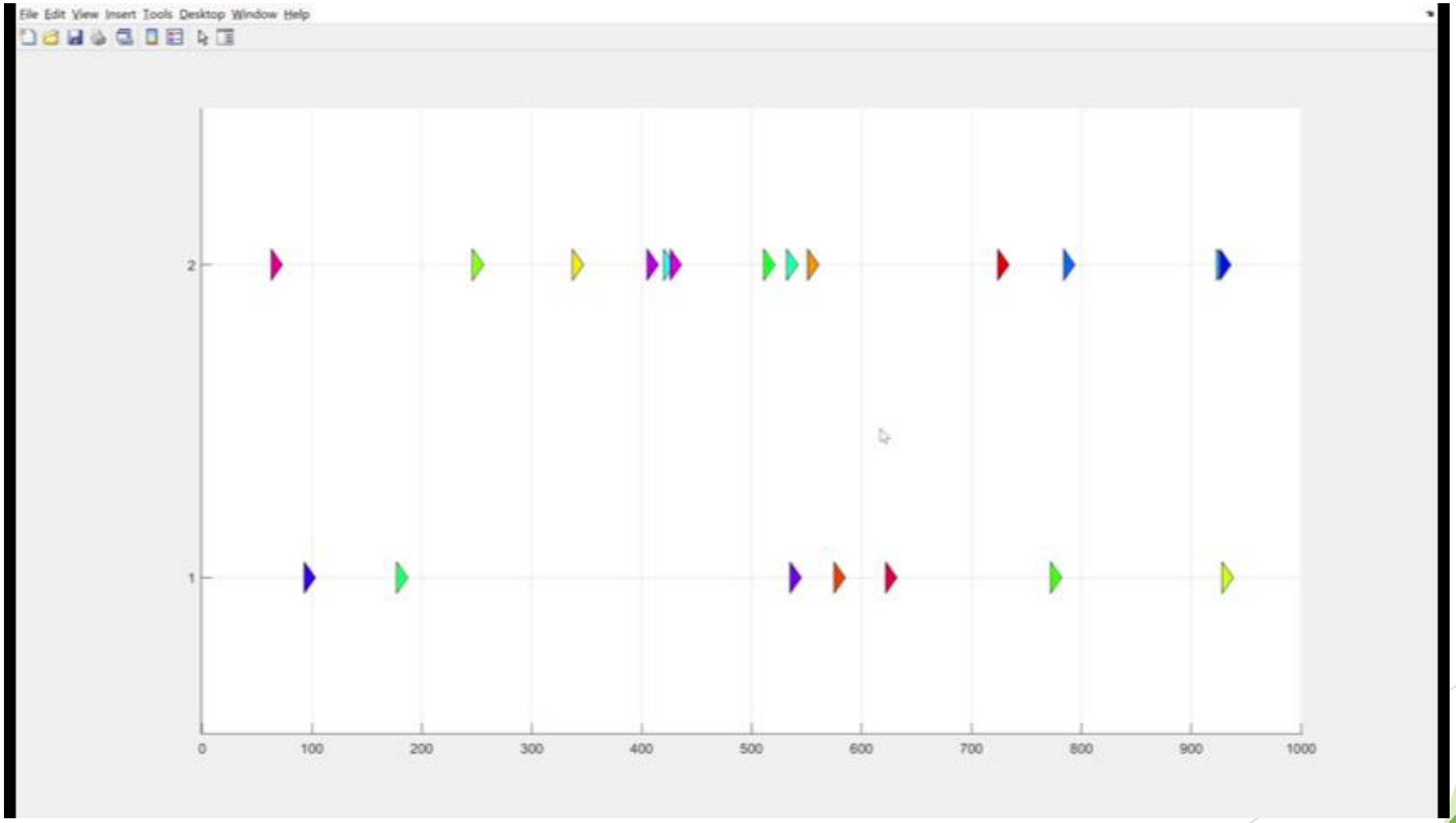
Two-Lane Traffic Simulation

Tommy Poek

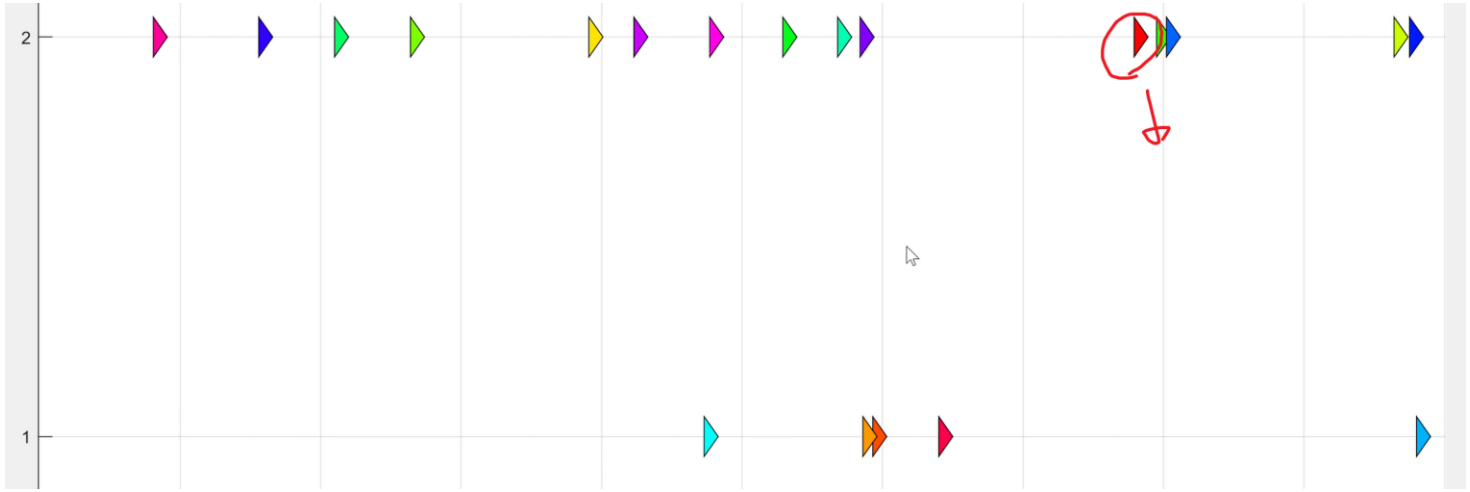
47192085

Intro – Visualization

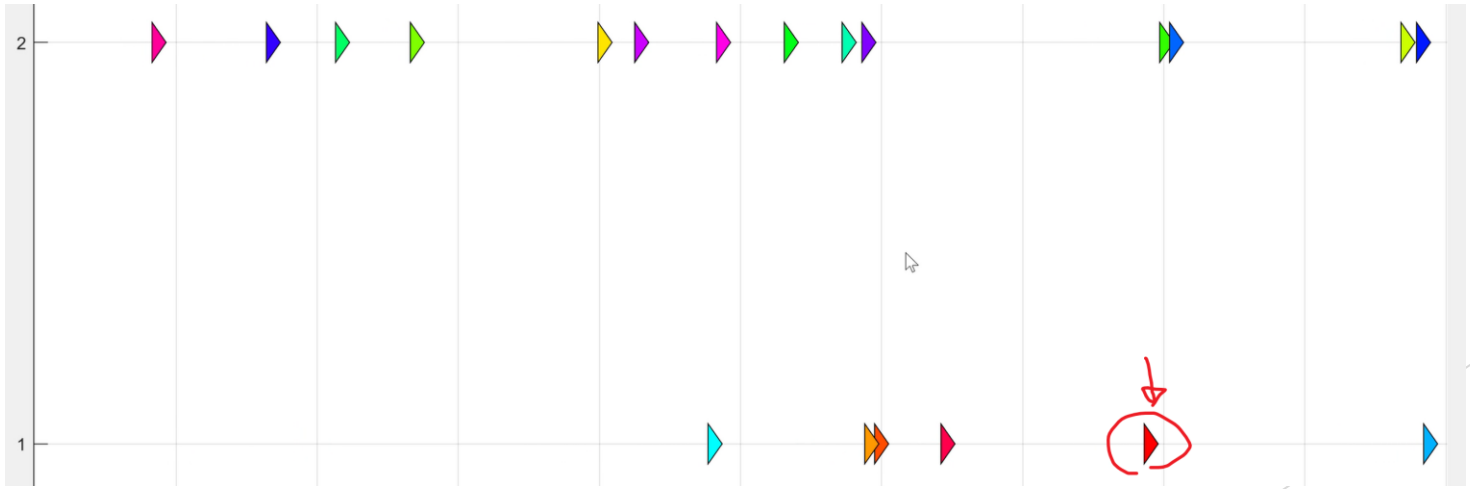


Intro – Visualization

- Step i

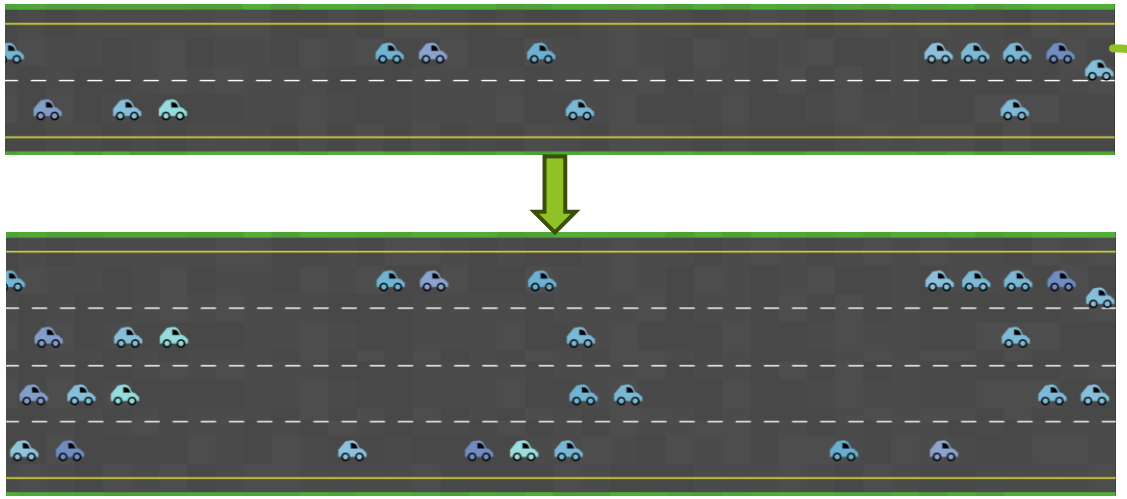


- Step i+1

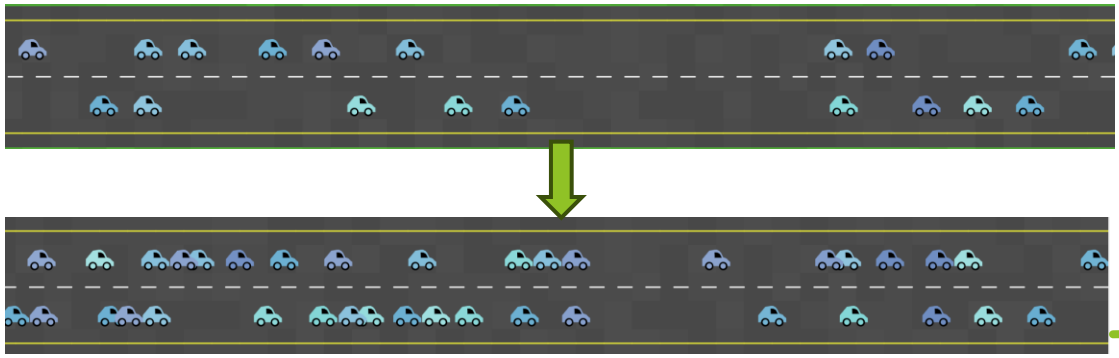


Intro - Scaling

- Scale up #lanes



- Scale up #cars



Fair benchmarking ←

Equal chances of lane change ←

Constant traffic density ←

Scale both #cars and #lanes

Intro - Scaling

- ~~Scale up #lanes~~

- Scale up #cars

Fair benchmarking ←

Equal chances of lane change ←

Constant traffic density ←

Scale #cars and lane_length

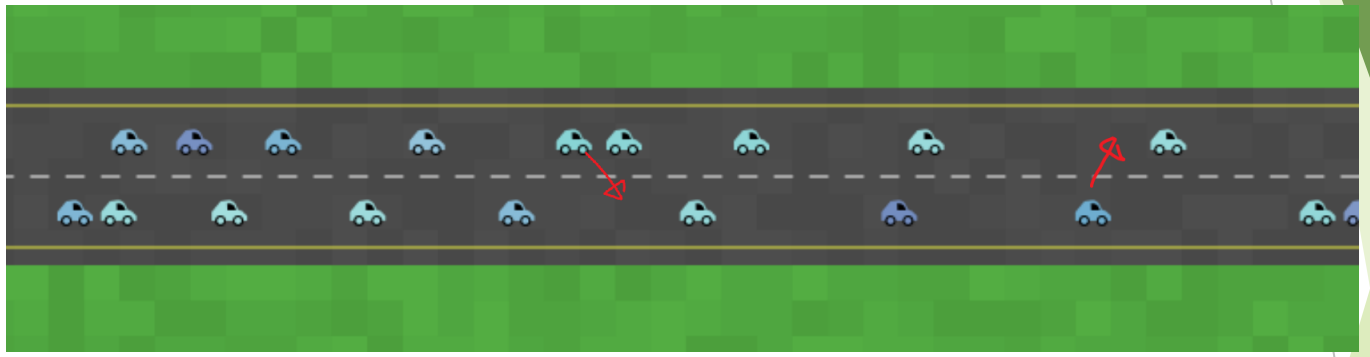
Simplified for GPU optimization

Intro – How the model works

- for each simulation step:

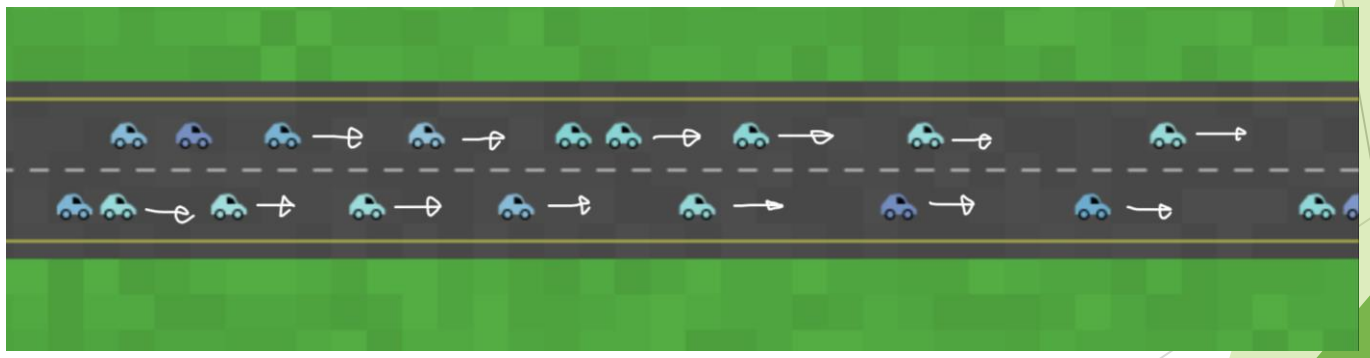
tryLaneChange

- // tryLaneChange
  - /*start the clock*/
  - determineTargetPosition()
  - tryLaneChange()
  - /*time the clock*/



driveForward

- // driveForward
  - /*start the clock*/
  - resolveCollisionsPerLane()
  - updateActualPosition()
  - /*time the clock*/



Intro – How the model works

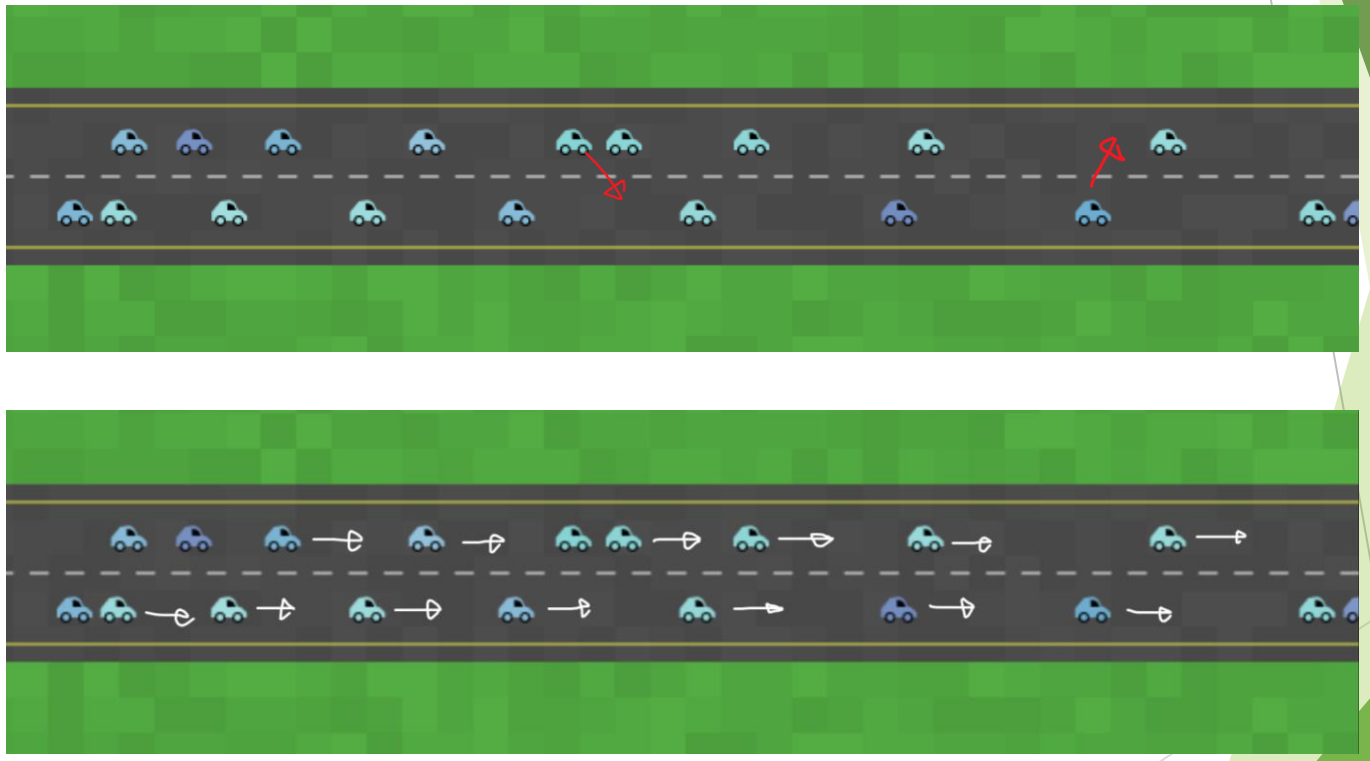
non-parallelizable

parallelize cars

parallelize lanes

for each simulation step:

- // tryLaneChange
  - /*start the clock*/
  - determineTargetPosition()
  - tryLaneChange()
  - /*time the clock*/
- // driveForward
  - /*start the clock*/
  - resolveCollisionsPerLane()
  - updateActualPosition()
  - /*time the clock*/

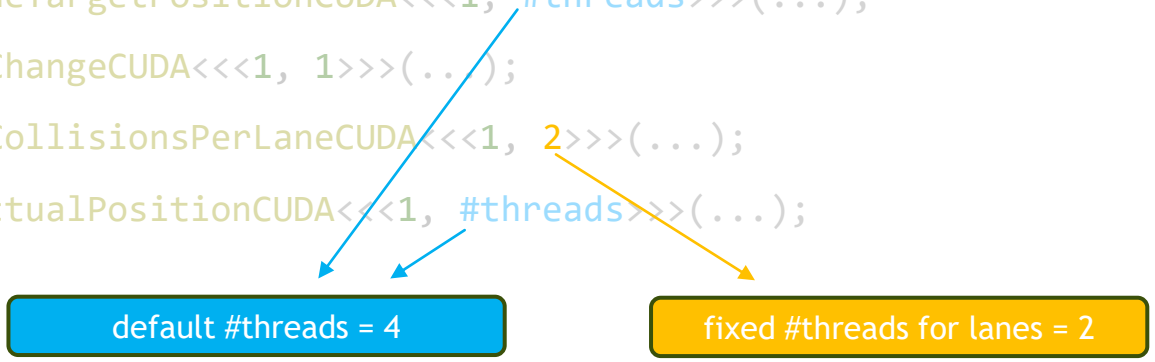


Implementations

- CPU
  - serial codes, same logic
- GPU -> CUDA thrust + manual malloc (hybrid) -> let's call it "GPU thrust"
  - `thrust::for_each(..., ..., DetermineTargetPosition());`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `thrust::for_each(..., ..., UpdateActualPosition());`
- GPU -> CUDA manual malloc (purely manual) -> let's call it "GPU manual"
  - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `updateActualPositionCUDA<<<1, #threads>>>(...);`

Implementations

- CPU
  - serial codes, same logic
- GPU -> CUDA thrust + manual malloc (hybrid) -> let's call it "GPU thrust"
  - `thrust::for_each(..., ..., DetermineTargetPosition());`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `thrust::for_each(..., ..., UpdateActualPosition());`
- GPU -> CUDA manual malloc (purely manual) -> let's call it "GPU manual"
  - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `updateActualPositionCUDA<<<1, #threads>>>(...);`



Implementations

- CPU
  - serial codes, same logic
- GPU -> CUDA thrust + manual malloc (hybrid) -> let's call it "GPU thrust"
  - `thrust::for_each(..., ..., DetermineTargetPosition());`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `thrust::for_each(..., ..., UpdateActualPosition());`
- GPU -> CUDA manual malloc (purely manual) -> let's call it "GPU manual"
  - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `updateActualPositionCUDA<<<1, #threads>>>(...);`

CUDA Nsight Compute profiling, #cars=500

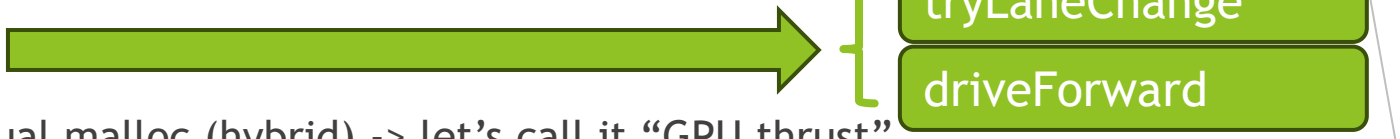
Function Name	De	Duration (1.10943e+09)	Ru (1.	Compute Throughput	Memory Throughput	# Registers	Grid Size	Block Size
determineTargetPositionCUDA	...	0.06		0.05	0.13	24	1...	4...
tryLaneChangeCUDA	...	11.74		0.91	0.91	48	1...	1...
resolveCollisionsPerLaneCUDA	...	0.18		0.08	0.08	40	1...	2...
updateActualPositionCUDA	...	0.05		0.04	0.12	30	1...	4...

default #threads = 4

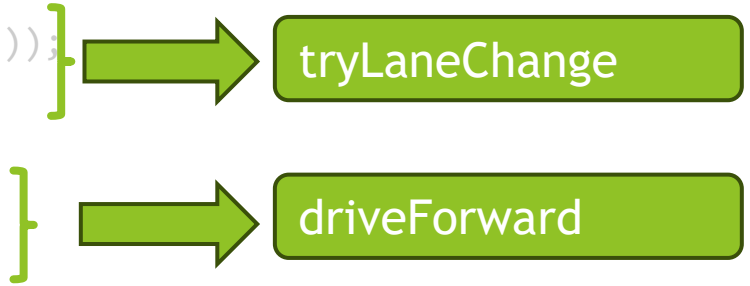
fixed #threads for lanes = 2

Implementations

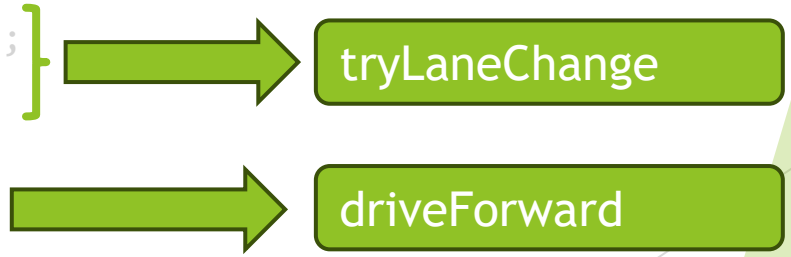
- CPU
  - serial codes, same logic



- GPU -> CUDA thrust + manual malloc (hybrid) -> let's call it "GPU thrust"
  - `thrust::for_each(..., ..., DetermineTargetPosition());`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `thrust::for_each(..., ..., UpdateActualPosition());`



- GPU -> CUDA manual malloc (purely manual) -> let's call it "GPU manual"
  - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
  - `tryLaneChangeCUDA<<<1, 1>>>(...);`
  - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
  - `updateActualPositionCUDA<<<1, #threads>>>(...);`



Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

GPU thrust vs CPU:

- tryLaneChange significantly optimized!!
- driveForward runs terribly longer!

Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

GPU thrust vs CPU:

- tryLaneChange significantly optimized!!
- driveForward runs terribly longer!

GPU manual vs CPU:

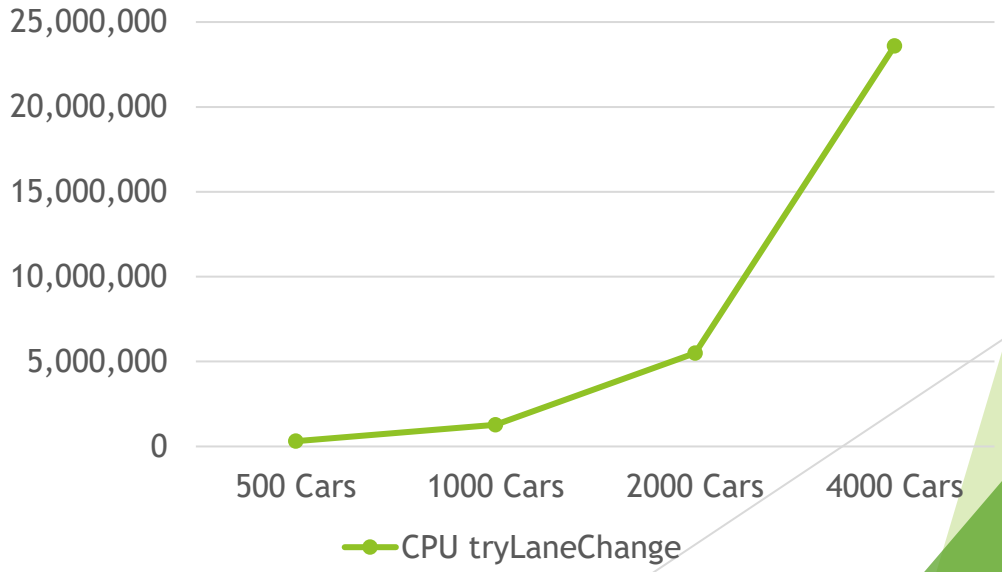
- both tryLaneChange and driveForward are optimized!!

Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

CPU tryLaneChange:

- Traffic size increases → Runtime explodes!

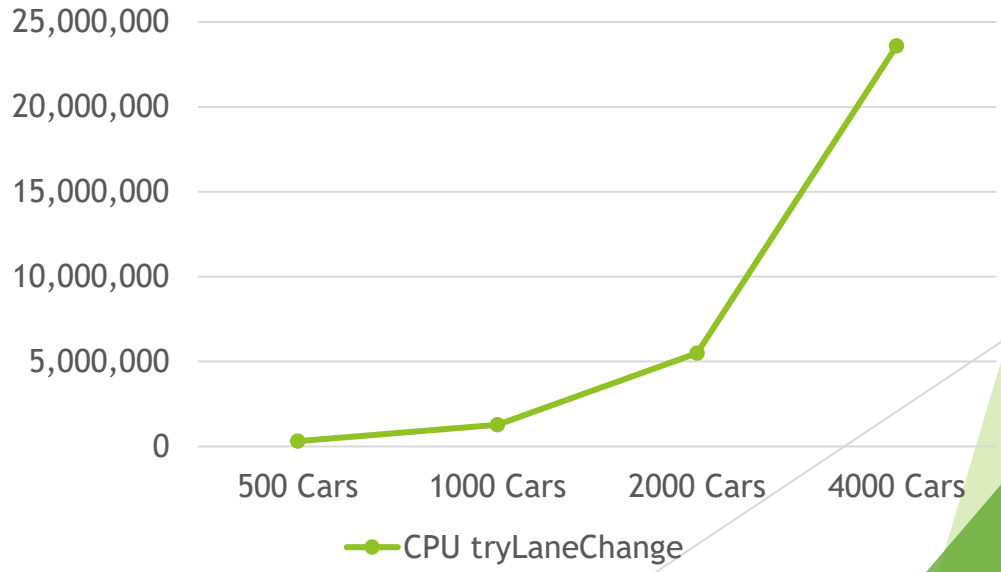


Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

	# lane changes	3,555	4,260	6,135	16,229
Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

CPU tryLaneChange:

- Traffic size increases → Runtime explodes!
- Successful lane changes adds #instructions

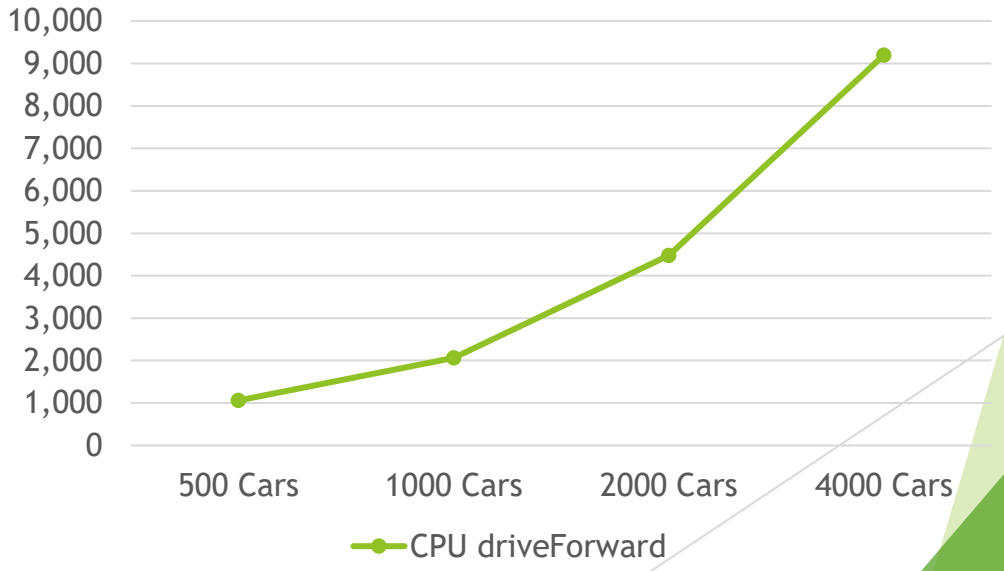


Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

CPU driveForward:

- Traffic size increases → Runtime increases linearly!

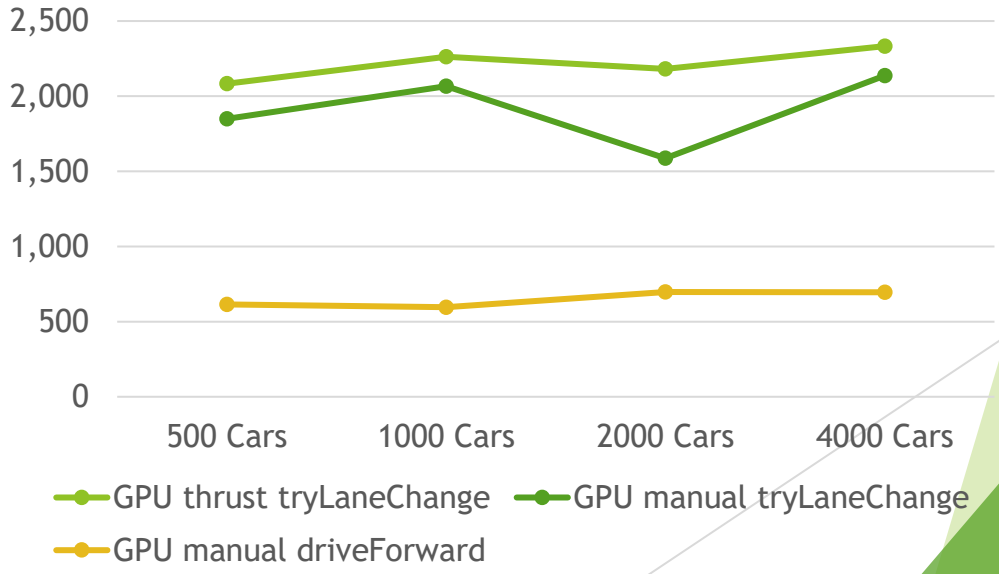


Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

GPU (thrust and manual) tryLaneChange:
and GPU manual driveForward:

- Traffic size increases → Runtime remains stable!

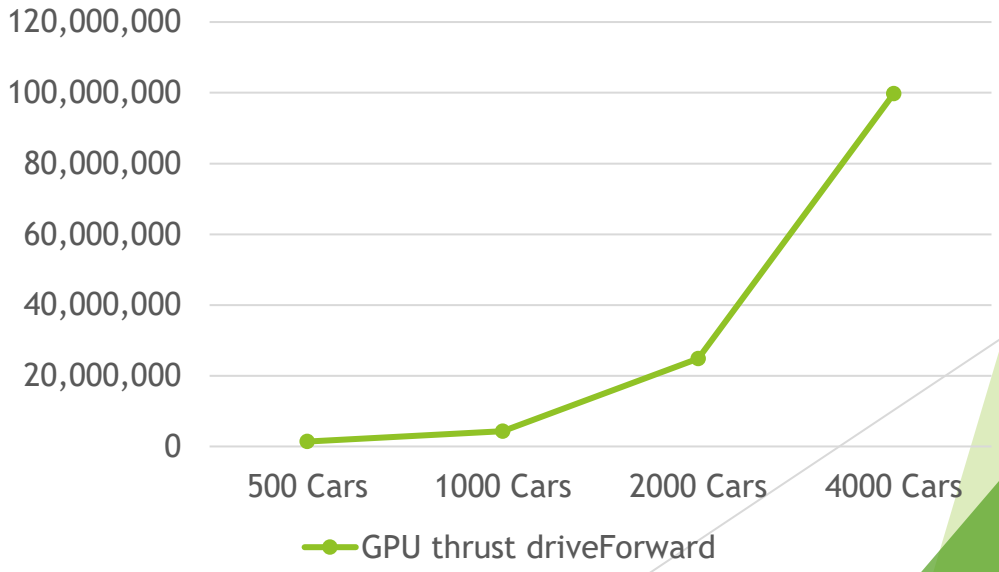


Benchmarking – CPU vs GPU-thrust vs GPU-manual all runtime measured in us

Implementation	Task	500 Cars	1000 Cars	2000 Cars	4000 Cars
CPU	tryLaneChange	309,860	1,267,526	5,507,635	23,597,986
	driveForward	1,060	2,061	4,480	9,199
GPU thrust	tryLaneChange	2,083	2,263	2,182	2,333
	driveForward	1,388,718	4,347,869	24,827,547	99,747,402
GPU manual	tryLaneChange	1,849	2,066	1,587	2,137
	driveForward	614	596	697	695

GPU thrust driveForward:

- Traffic size increases → Runtime explodes!



What I’ve learnt

- CPU with increasing traffic size:
  - tryLaneChange runtime explodes! ← more successful lane changes, more instructions
  - driveForward runs linearly slower ← #instructions proportional to problem size

What I've learnt

- CPU with increasing traffic size:
  - tryLaneChange runtime explodes! ← more successful lane changes, more instructions
  - driveForward runs linearly slower ← #instructions proportional to problem size
- from CPU to GPU with increasing traffic size:
  - tryLaneChange (both thrust and manual versions) get optimized and remain stable ← parallelization
  - driveForward (manual version) get optimized and remain stable ← parallelization
    - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
    - `tryLaneChangeCUDA<<<1, 1>>>(...);`
    - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
    - `updateActualPositionCUDA<<<1, #threads>>>(...);`

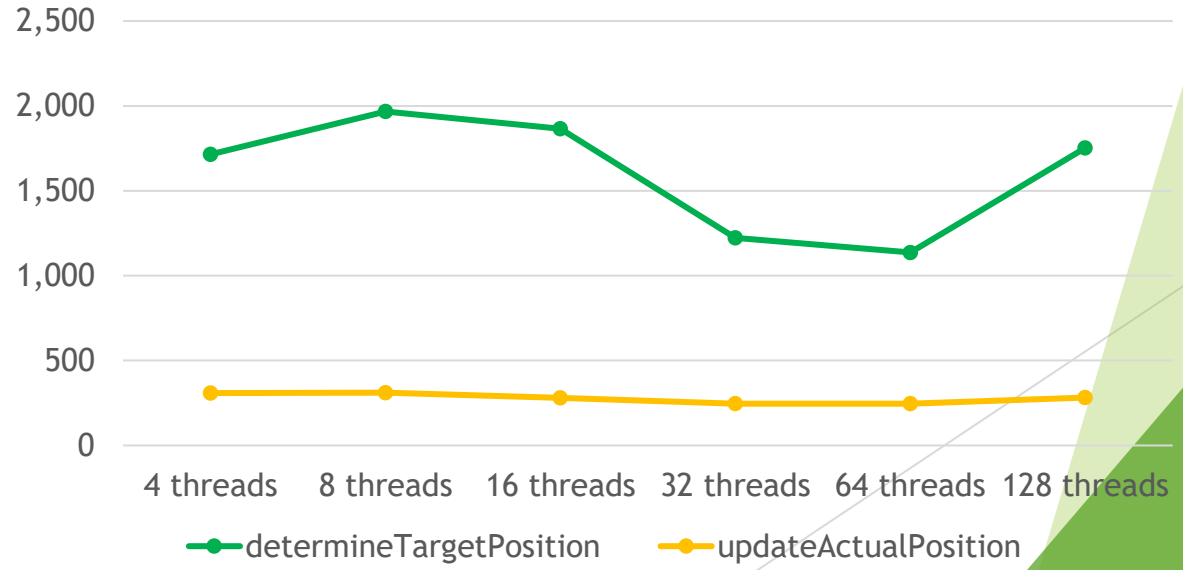
What I've learnt

- CPU with increasing traffic size:
  - tryLaneChange runtime explodes! ← more successful lane changes, more instructions
  - driveForward runs linearly slower ← #instructions proportional to problem size
- from CPU to GPU with increasing traffic size:
  - tryLaneChange (both thrust and manual versions) get optimized and remain stable ← parallelization
  - driveForward (manual version) get optimized and remain stable ← parallelization
    - `determineTargetPositionCUDA<<<1, #threads>>>(...);`
    - `tryLaneChangeCUDA<<<1, 1>>>(...);`
    - `resolveCollisionsPerLaneCUDA<<<1, 2>>>(...);`
    - `updateActualPositionCUDA<<<1, #threads>>>(...);`
  - driveForward (thrust version) runs much slower! ... too much abstraction in thrust
    - `thrust::for_each(carsDevice.begin(), carsDevice.end(), UpdateActualPosition());`

Benchmarking – GPU manual malloc – different #threads all runtime measured in us

Tasks (fixed #cars=2000)	#threads=4	#threads=8	#threads=16	#threads=32	#threads=64	#threads=128
determineTargetPosition<<<1, #threads>>>	1,715	1,967	1,866	1,222	1,137	1,753
updateActualPosition<<<1, #threads>>>	309	311	280	246	246	282

determineTargetPosition: best #threads = 64
updateActualPosition: Not affected by #threads



What I’ve learnt

- With fixed #cars=2000, as #threads increases →
  - determineTargetPosition optimized! (till #threads=64) ←
    - Arithmetic instruction: `car.TargetPosition = car.Position + car.TargetSpeed;`
  - No effects on updateActualPosition ←
    - Too simple instruction: `car.Position = car.TargetPosition;`

Thank you!

Tommy Poek

47192085