Parallelizing a Real-Time Steering Simulation for Computer Games with OpenMP

Bjoern Knafla and Claudia Leopold
University of Kassel (Germany)
Steering a flock of birds
Dienstag, 11. September 2007
Bird
Bird
Boid
Dienstag, 11. September 2007
Dienstag, 11. September 2007
Dienstag, 11. September 2007
Dienstag, 11. September 2007
Outline
Outline

1. Steering behaviors
Outline

1. Steering behaviors
2. OpenSteerDemo
Outline

1. Steering behaviors
2. OpenSteerDemo
3. Bad parallelization
Outline

1. Steering behaviors
2. OpenSteerDemo
3. Bad parallelization
4. Parallelization that works
Outline

1. Steering behaviors
2. OpenSteerDemo
3. Bad parallelization
4. Parallelization that works
5. Performance
Outline

1. Steering behaviors
2. OpenSteerDemo
3. Bad parallelization
4. Parallelization that works
5. Performance
6. Summary
Steering behaviors
Alignment
OpenSteerDemo
OpenSteerDemo

Testbed for the open source library OpenSteer from Craig W. Reynolds

9 Dienstag, 11. September 2007
OpenSteer Demo

Testbed for the open source library OpenSteer from Craig W. Reynolds

Simulates steering behavior of agents
Testbed for the open source library OpenSteer from Craig W. Reynolds
Simulates steering behavior of agents
Game-like C++ real-time application
Testbed for the open source library OpenSteer from Craig W. Reynolds
Simulates steering behavior of agents
Game-like C++ real-time application
OpenGL
OpenSteerDemo

Testbed for the open source library OpenSteer from Craig W. Reynolds

Simulates steering behavior of agents

Game-like C++ real-time application

OpenGL

http://opensteer.sourceforge.net
Main loop
Main loop

Input → Update → Output
Main loop

Input → Update → Graphics
Input stage

Time step $t$

Input $\rightarrow$ Update $\rightarrow$ Graphics
Update stage

Time step $t$

Input → Update → Graphics
Graphics stage

Time step $t$

Input → Update → Graphics
Next main loop cycle

Time step $t+1$
Update stage

Input → Update → Graphics

Dienstag, 11. September 2007
Agent update

Update
agent

Agent
state
read + write

Neighbors
read + write

Random.
read + write

Graphics
write

references
Time step $t$

In state $t-1$

- Update agent 0
- Update agent 1
- Update agent 2
- ... Update agent $n$

Sequential agent update order

Dienstag, 11. September 2007
Time step $t$

In state $t-1$

Update agent 0
Update agent 1
Update agent 2
... Update agent $n$

Sequential agent update order

Neighbors

In state $t-1$
Time step $t$
Time step $t$

Sequential agent update order

In state $t$

- Update agent 0
- Update agent 1

In state $t-1$

- Update agent 1
- Update agent 2
- ... (ellipses)
- Update agent $n$
Time step $t$
Time step $t$

Sequential agent update order

- Update agent 0
- Update agent 1
- Update agent 2
- ... Update agent n

In state $t$

Neighbors

In state $t-1$

In state ?

Dienstag, 11. September 2007
Bad parallelization
Questionable reliability and correctness
No speedup
No speedup
Main parallelization problems
Race conditions
Race conditions

Global variables, deep inheritance hierarchies, strongly interdependent classes
Race conditions

Global variables, deep inheritance hierarchies, strongly interdependent classes

Non-determinism
Race conditions

Global variables, deep inheritance hierarchies, strongly interdependent classes

Non-determinism

Order of agent updates and random numbers
Race conditions

Global variables, deep inheritance hierarchies, strongly interdependent classes

Non-determinism

Order of agent updates and random numbers

Non-thread-safe functions
Parallelization that works
Guiding ideas
Guiding ideas

Update order independent simulation
Guiding ideas

Update order independent simulation

Agent modification only dependent on its steering vector
Guiding ideas

Update order independent simulation

Agent modification only dependent on its steering vector

Minimize synchronization
Guiding ideas

Update order independent simulation
Agent modification only dependent on its steering vector
Minimize synchronization
Finish parallel processing before graphics stage
Guiding ideas

Update order independent simulation
Agent modification only dependent on its steering vector
Minimize synchronization
Finish parallel processing before graphics stage
Interfaces for explicit context
Refactorization
Split agent update

Agent update

Sim. agent

Modify agent

Agent update order

Dienstag, 11. September 2007
Simulate agent stage

Steering Vector
Random.
Agent state
Agent public
Agent private

Sim. agent
read + write

read
Agent state

read
Neighbors

write
Render-Feeder

references
Simulate agent stage

- Sim. agent
  - Agent public
  - Agent private
  - Random.

- Steering Vector
- Agent state
- Agent public
- Agent private

- Neighbors
- Render-Feeder

- read + write
- const
- read
- const
- read
- const
- write

Dienstag, 11. September 2007
Modify agent stage

Steering Vector
Random.
Agent state
Agent private

Modify agent

Agent state

Neighbors
Render-Feeder

read + write
references

read
Agent public
Modify neighbor data structure

Steering Vector
Random.
Agent state
Agent private

read
Agent public
references
write
Agent state
Neighbors

Modify neighbors

Render-Feeder
Refactor update stage

Input -> Update -> Output

Dienstag, 11. September 2007
Split update stage

Sub-stages: simulation and modification
Old update stage

Sequential processing order

Update agent

Sim. agent 0
Modify agent 0
Sim. agent 1
Modify agent 1
Sim. agent n
Modify agent n

...
Time step $t$

Sequential processing order

Agents in state $t-1$

Sim. agent 0

Sim. agent 1

Sim. agent n

Modify agent 0

Modify agent 1

Modify agent n
Time step t

Sequential processing order

Agent in state t

Sim. agent 0
Modify agent 0
Sim. agent 1
Modify agent 1

Agents in state t-1

Sim. agent n
Modify agent n
New update stage

Sequential processing order

Simulation sub-stage

Sim. agent 0  Sim. agent 1  ...  Sim. agent n

Modification sub-stage

Modify agent 0  Modify agent 1  ...  Modify agent n

Dienstag, 11. September 2007
Time step $t$

Agents in public state $t-1$

Sequential processing order

- Sim. agent 0
- Sim. agent 1
- ... Sim. agent $n$
- Modify agent 0
- Modify agent 1
- ... Modify agent $n$
Time step $t$

Agents in public state $t-1$

Sim. agent 0
Sim. agent 1
... Sim. agent n

Modify agent 0
Modify agent 1
... Modify agent n

Sequential processing order
Time step $t$

Agents in public state $t-1$

Sequential processing order

Sim. agent 0  Sim. agent 1  ...  Sim. agent $n$  Modify agent 0  Modify agent 1  ...  Modify agent $n$
Time step $t$

Agents in public state $t-1$

Sequential processing order

Each agent is modified based solely on its own state
Parallelization
Parallel update stage

Simulation

Modification

Update neighbors

Possibly parallel
Parallel simulation sub-stage

Simulation

Parallel simulation

Modification

Update neighbors

Possibly parallel
Parallel modification
sub-stage

Simulation

Modification

Update neighbors
Possibly parallel

Parallel barrier
Parallel barrier
Parallel barrier
Parallel barrier
Parallel modification
sub-stage

Simulation

Modification

Update neighbors

Possibly parallel
Performance
Test-computer

Dual-processor dual-core 2 GHz AMD Opteron with 2 GB Ram

2x Nvidia 7800 GTX graphics cards in SLI mode

Linux OS
Max speedups
Max speedups

Comparison of OpenSteerDemo flock of birds simulation with 4 threads with OpenMP-disabled version
Max speedups

Comparison of OpenSteerDemo flock of birds simulation with 4 threads with OpenMP-disabled version

Whole application: 2.84
Max speedups

Comparison of OpenSteerDemo flock of birds simulation with 4 threads with OpenMP-disabled version

Whole application: 2.84

Update stage: 3.54
Summary
Refactored first
Refactored first

Simplicity!
Refactored first

Simplicity!

High-level design to min. synchronization
Refactored first

Simplicity!

High-level design to min. synchronization

Explicit context
Refactored first

Simplicity!

High-level design to min. synchronization

Explicit context

Deferred computation
Refactored first
Simplicity!
High-level design to min. synchronization
Explicit context
Deferred computation
Identified slices for data-parallel computation
Thank you!