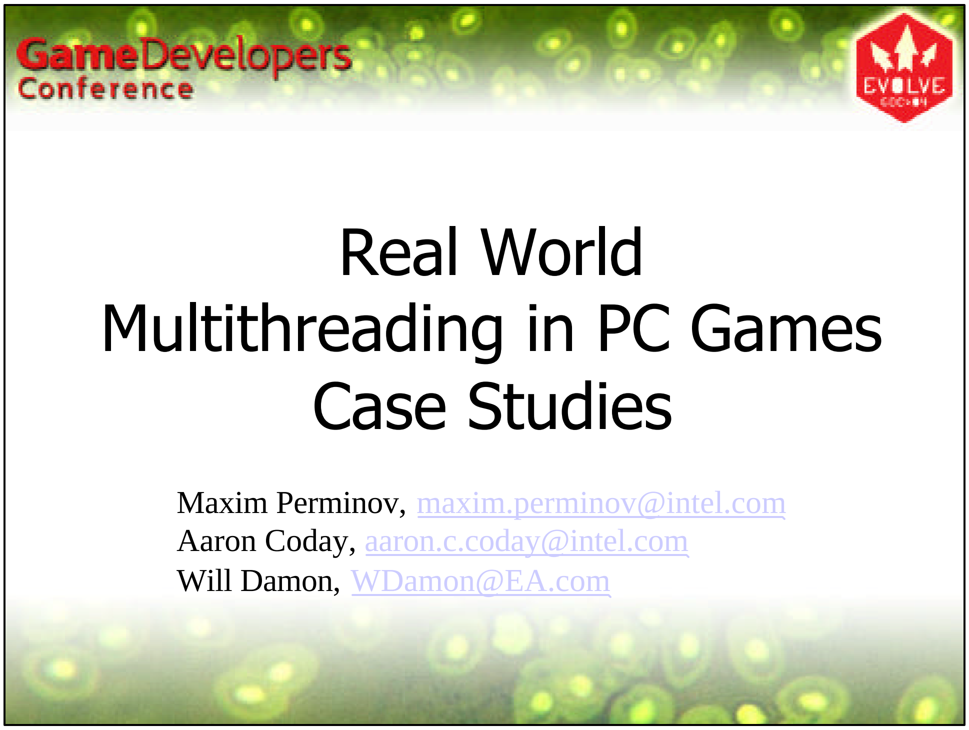
<< RETURN



GameDevelopers Conference
EVOLVE GDC•04

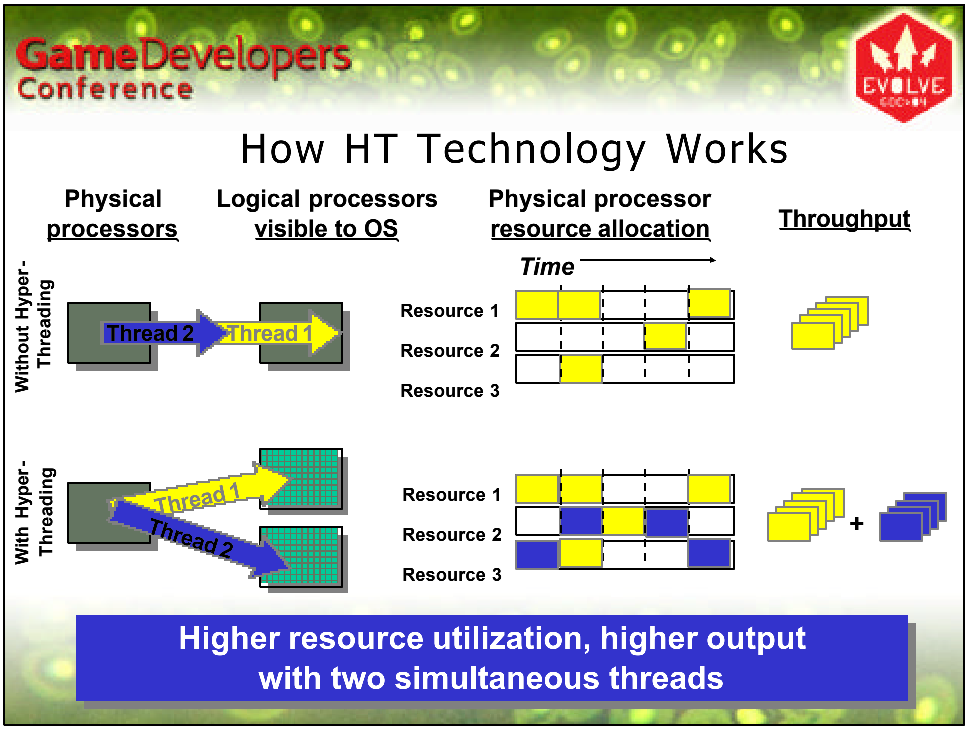
Real World Multithreading in PC Games Case Studies

Maxim Perminov, maxim.perminov@intel.com
Aaron Coday, aaron.c.coday@intel.com
Will Damon, WDamon@EA.com

GameDevelopers Conference
EVOLVE GDC•04

Agenda

- Hyper-Threading Technology Review
- Multithreading Challenges & Strategies for Games
- Case Studies
  - Lego/Argonaut *"Bionicle"*
  - Codemasters/SixByNine *"Colin McRae Rally 4"*
- Summary



GameDevelopers Conference

EVOLVE

How HT Technology Works

Physical processors

Logical processors visible to OS

Physical processor resource allocation

Throughput

Time

Without Hyper-Threading

Thread 2

Thread 1

Resource 1

Resource 2

Resource 3

With Hyper-Threading

Thread 1

Thread 2

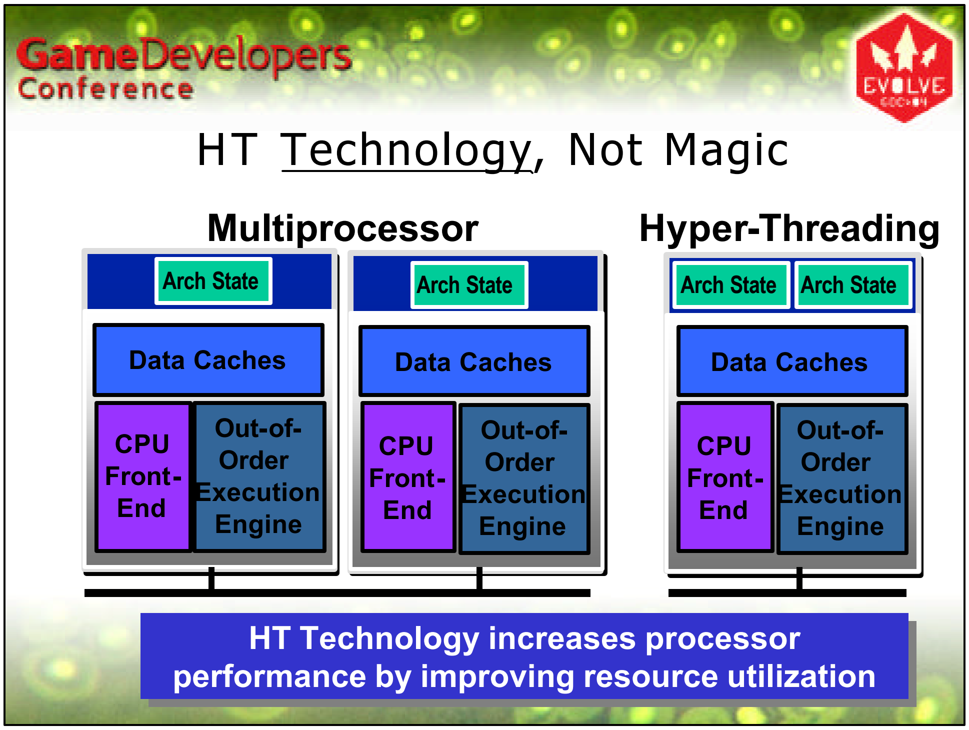
Resource 1

Resource 2

Resource 3

+

Higher resource utilization, higher output with two simultaneous threads



GameDevelopers Conference

EVOLVE

HT Technology, Not Magic

Multiprocessor

Arch State

Data Caches

CPU Front-End

Out-of-Order Execution Engine

Arch State

Data Caches

CPU Front-End

Out-of-Order Execution Engine

Hyper-Threading

Arch State

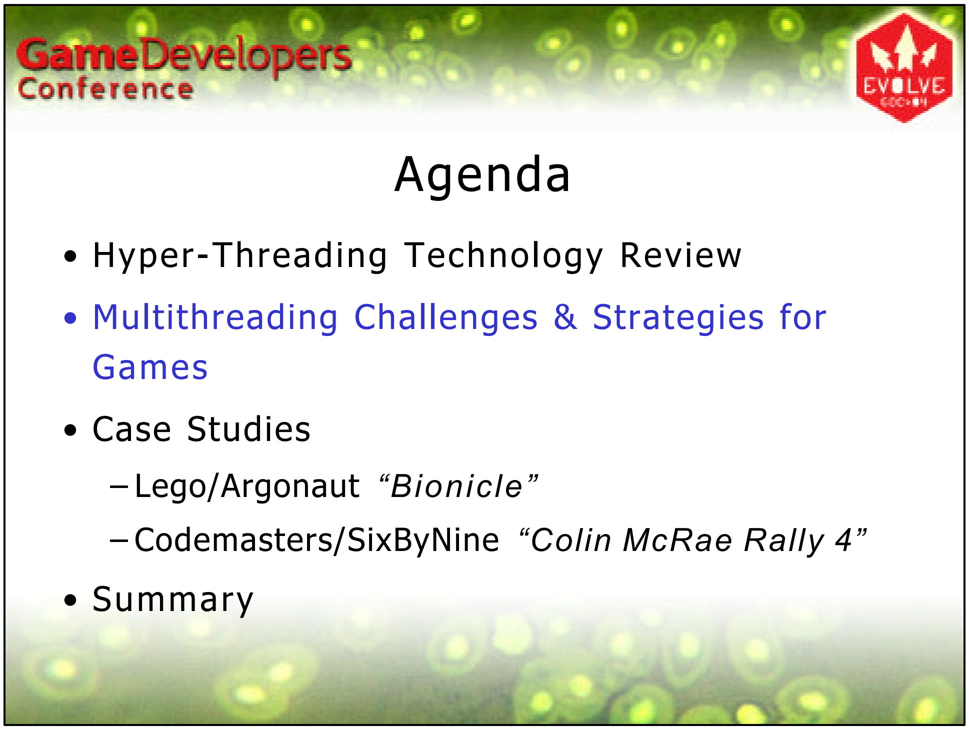
Arch State

Data Caches

CPU Front-End

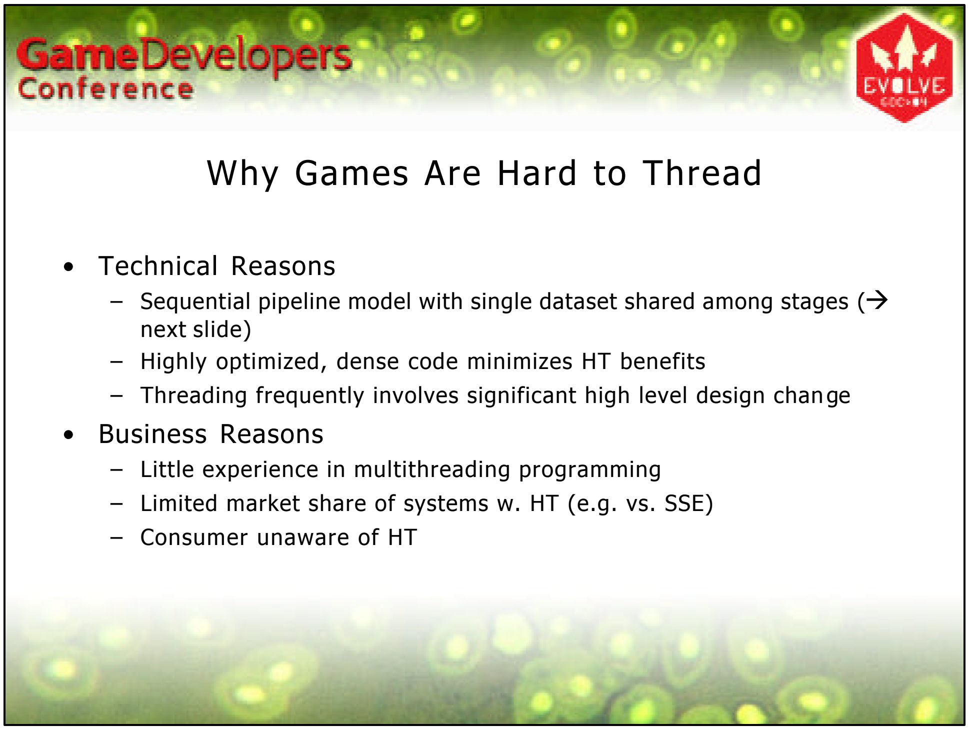
Out-of-Order Execution Engine

HT Technology increases processor performance by improving resource utilization


GameDevelopers Conference
EVOLVE

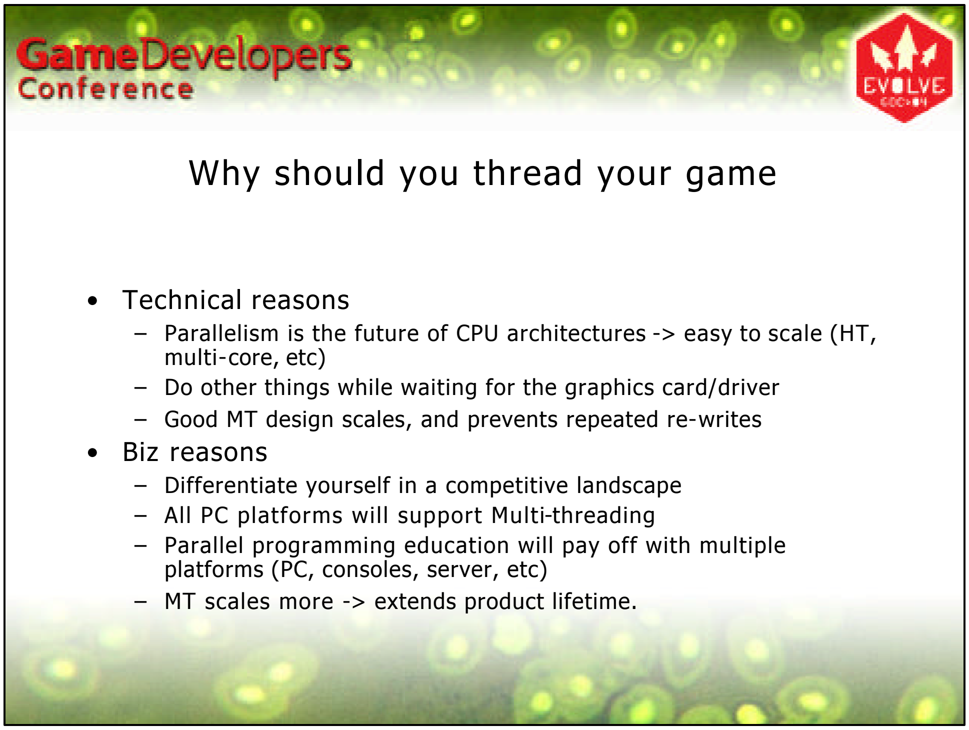
Agenda

- Hyper-Threading Technology Review
- Multithreading Challenges & Strategies for Games
- Case Studies
  - Lego/Argonaut *“Bionicle”*
  - Codemasters/SixByNine *“Colin McRae Rally 4”*
- Summary


GameDevelopers Conference
EVOLVE

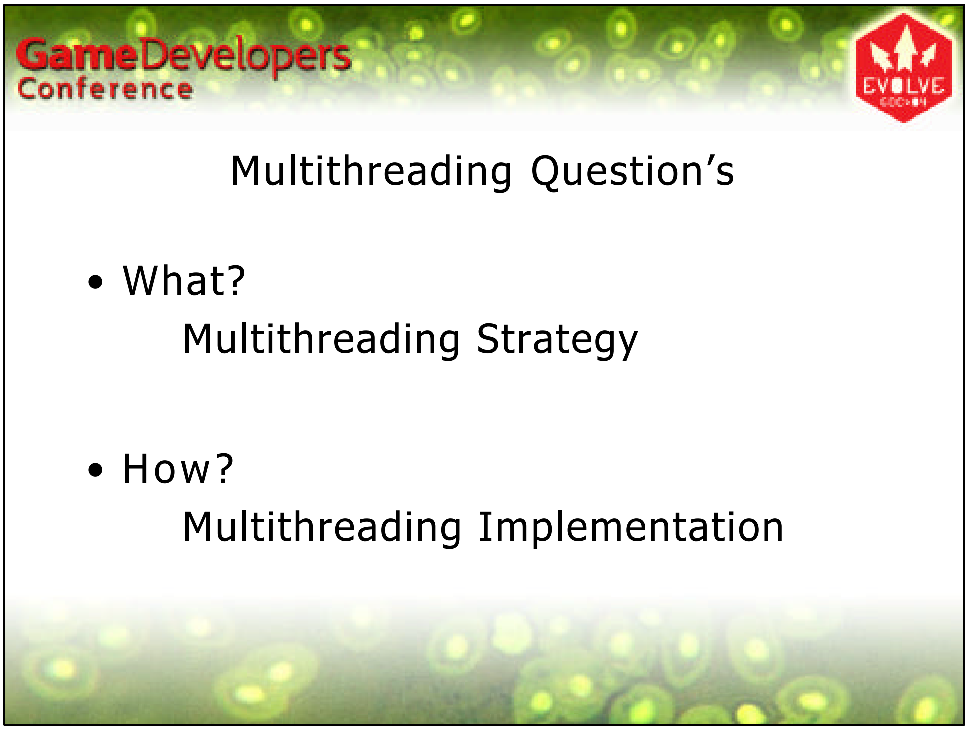
Why Games Are Hard to Thread

- Technical Reasons
  - Sequential pipeline model with single dataset shared among stages (→ next slide)
  - Highly optimized, dense code minimizes HT benefits
  - Threading frequently involves significant high level design change
- Business Reasons
  - Little experience in multithreading programming
  - Limited market share of systems w. HT (e.g. vs. SSE)
  - Consumer unaware of HT



Why should you thread your game

- Technical reasons
  - Parallelism is the future of CPU architectures -> easy to scale (HT, multi-core, etc)
  - Do other things while waiting for the graphics card/driver
  - Good MT design scales, and prevents repeated re-writes
- Biz reasons
  - Differentiate yourself in a competitive landscape
  - All PC platforms will support Multi-threading
  - Parallel programming education will pay off with multiple platforms (PC, consoles, server, etc)
  - MT scales more -> extends product lifetime.



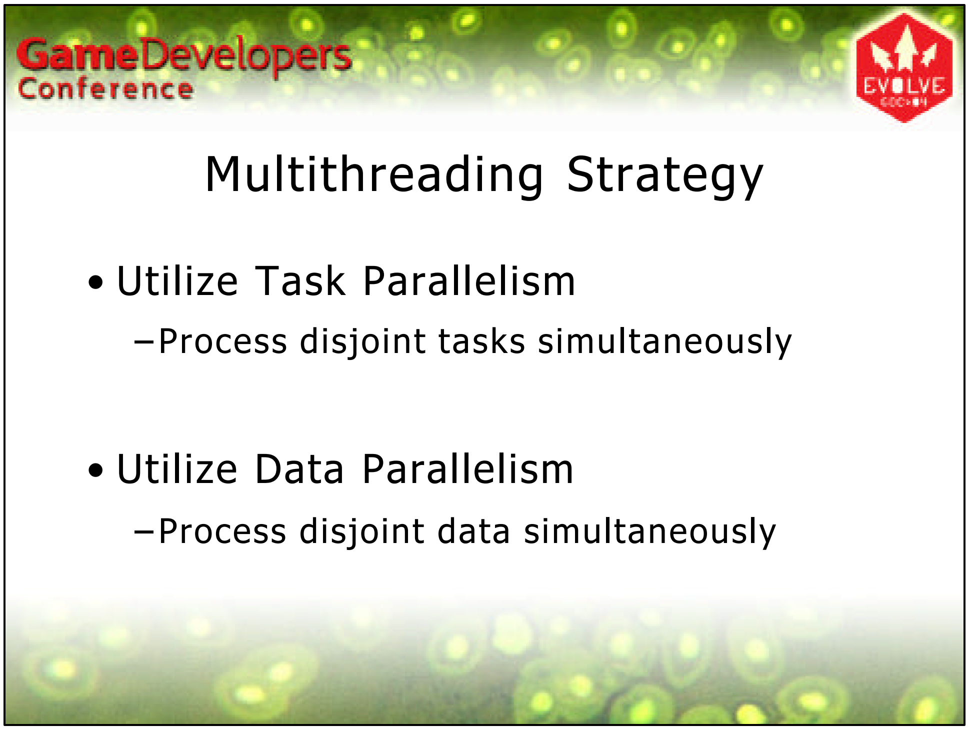
Multithreading Question’s

- What?

  Multithreading Strategy

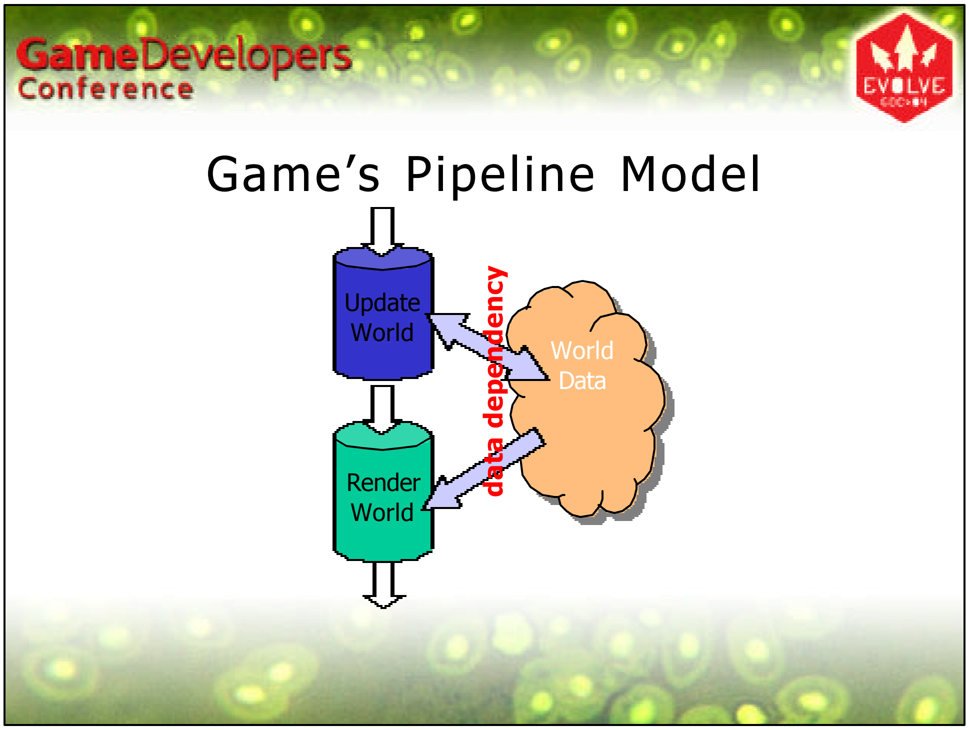
- How?

  Multithreading Implementation



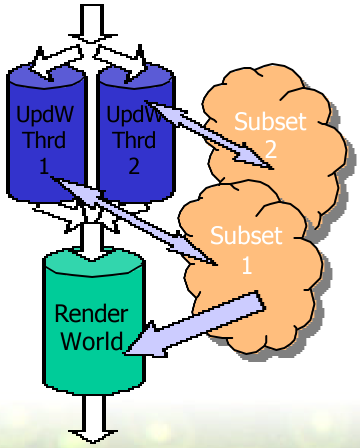
Multithreading Strategy

- Utilize Task Parallelism
  - Process disjoint tasks simultaneously
- Utilize Data Parallelism
  - Process disjoint data simultaneously



Game’s Pipeline Model

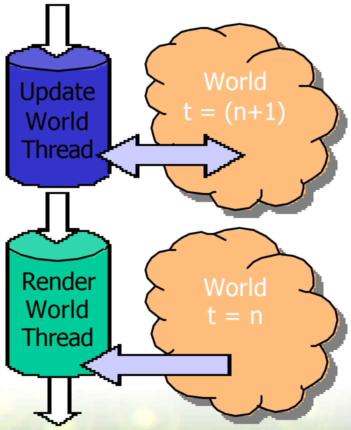
Data Parallelism in Games



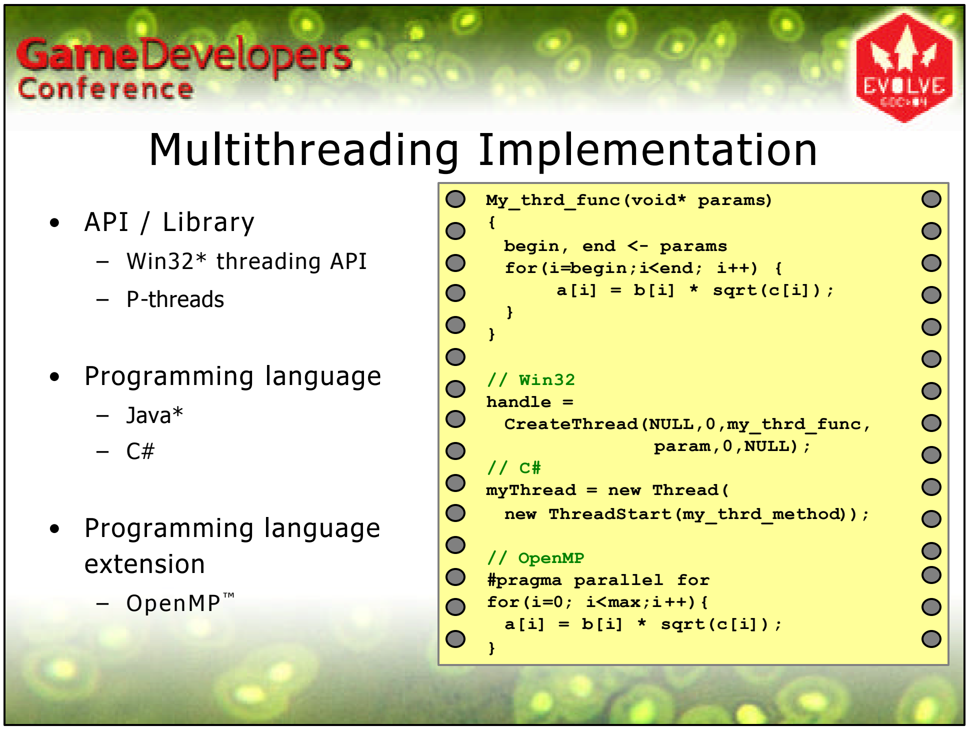
- Execute tasks on secondary thread
  - Audio processing
  - Networking (including VoIP)
  - Particle Systems and other graphics effects
  - Physics, AI (also in Java* / C#)
  - Content (speculative) loading & unpacking
- Multithread Procedural Content creation
  - Geometry, Textures, Environment, etc…
- Threading Potential
  - Good for CPU bound games
  - Easy to implement

*Other names and brands may be claimed as the property of others

Task Parallelism in Games



- Multithread whole 3D Graphics Pipeline
  - Thread 1 = Render Frame (n)
  - Thread 2 = Update Frame (n+1)
- Threading potential
  - Good for GPU bound games
  - Difficult to implement due to dependencies, but not impossible



GameDevelopers Conference

EVOLVE

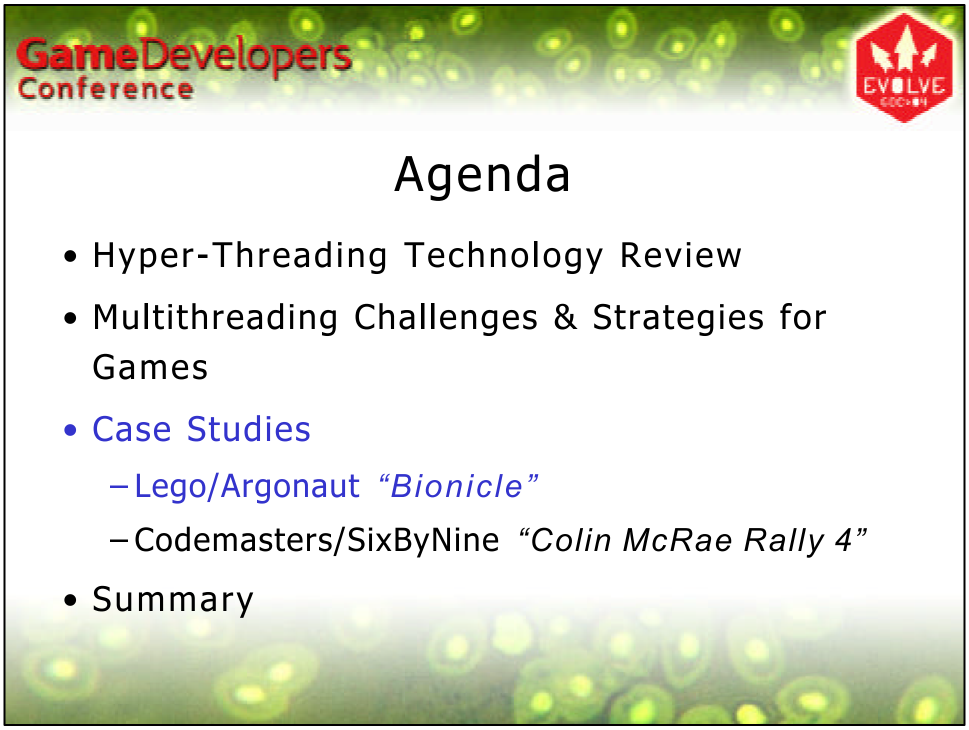
Multithreading Implementation

- API / Library
  - Win32* threading API
  - P-threads
- Programming language
  - Java*
  - C#
- Programming language extension
  - OpenMP™

```
My_thrd_func(void* params)
{
  begin, end <- params
  for(i=begin;i<end; i++) {
        a[i] = b[i] * sqrt(c[i]);
  }
}

// Win32
handle =
  CreateThread(NULL,0,my_thrd_func,
                 param,0,NULL);
// C#
myThread = new Thread(
  new ThreadStart(my_thrd_method));

// OpenMP
#pragma parallel for
for(i=0; i<max;i++){
  a[i] = b[i] * sqrt(c[i]);
}
```



GameDevelopers Conference

EVOLVE

Agenda

- Hyper-Threading Technology Review
- Multithreading Challenges & Strategies for Games
- Case Studies
  - Lego/Argonaut *“Bionicle”*
  - Codemasters/SixByNine *“Colin McRae Rally 4”*
- Summary

GameDevelopers Conference

BIONICLE THE GAME

LEGO

ARGONAUT

EVOLVE GDC•04

Case Study 2: LEGO*/Argonaut* *Bionicle**

- 3rd person Action Adventure
  - Based on successful LEGO toy franchise
  - Play as Toa in struggle of good vs. evil in the world of Mata Nui

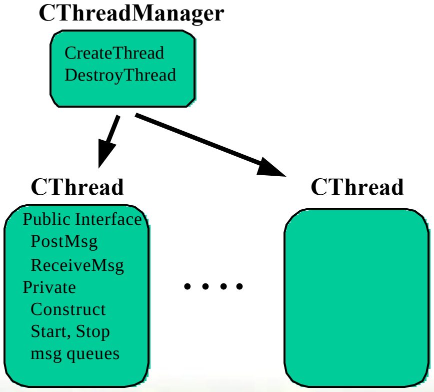
*Other names and brands may be claimed as the property of others

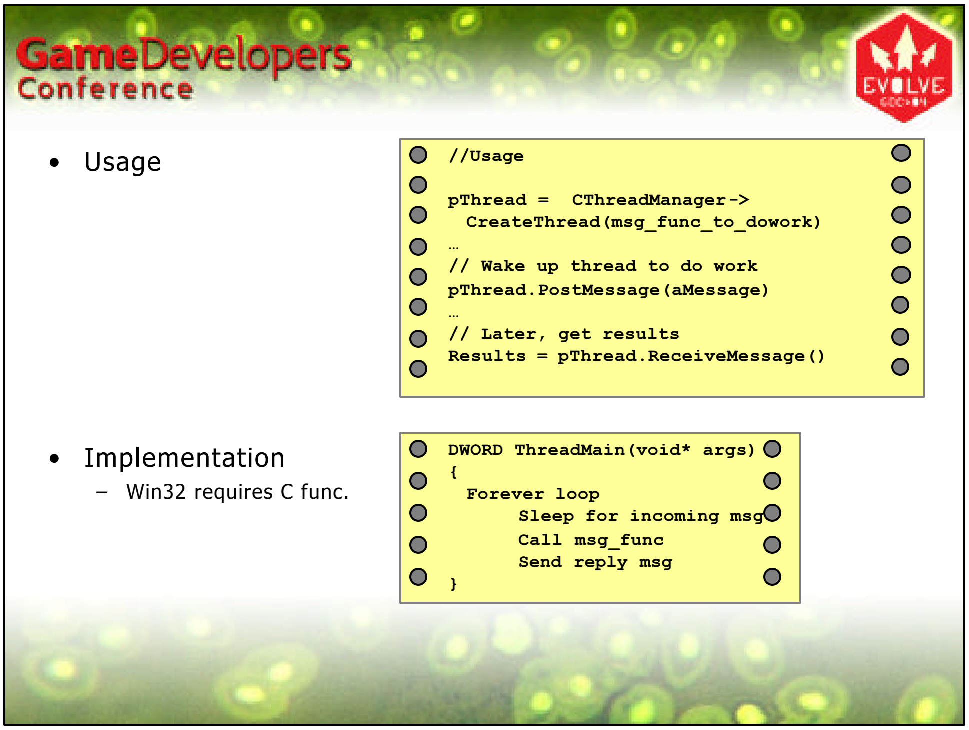
GameDevelopers Conference

EVOLVE GDC•04

Thread System

- Problem: Need a Threading system that is easy to use, object oriented and cross platform
- Solution: Roll our own Thread classes and message passing model.
  - Pros: Max flexibility, full control, and platform indep.
  - Cons: Harder up front, less supporting tools
- CThreadManager
  - Controls lifetime
- CThread
  - Wraps Win32 nitty gritty





GameDevelopers Conference

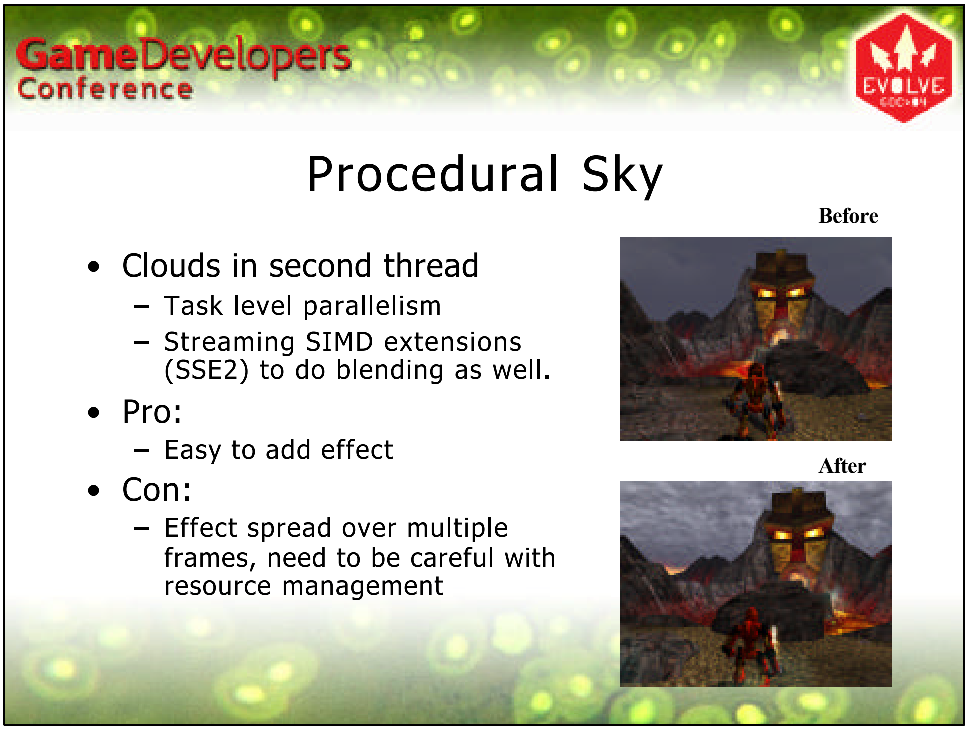
- Usage

```
//Usage

pThread =  CThreadManager->
  CreateThread(msg_func_to_dowork)
…
// Wake up thread to do work
pThread.PostMessage(aMessage)
…
// Later, get results
Results = pThread.ReceiveMessage()
```

- Implementation
  - Win32 requires C func.

```
DWORD ThreadMain(void* args)
{
  Forever loop
        Sleep for incoming msg
        Call msg_func
        Send reply msg
}
```



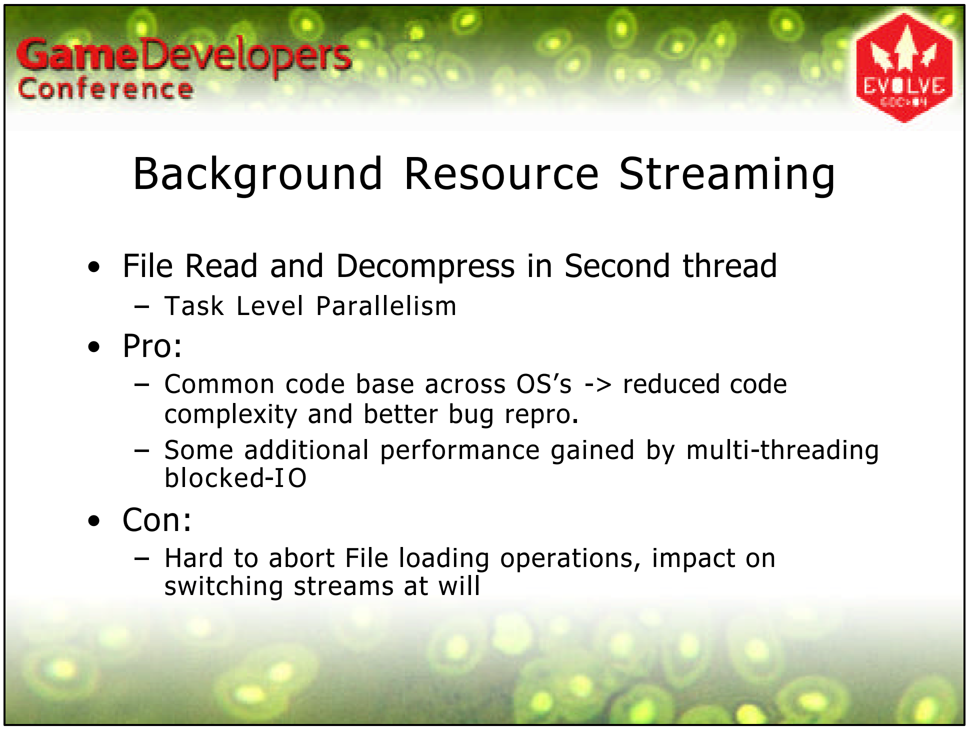
GameDevelopers Conference

Procedural Sky

- Clouds in second thread
  - Task level parallelism
  - Streaming SIMD extensions (SSE2) to do blending as well.
- Pro:
  - Easy to add effect
- Con:
  - Effect spread over multiple frames, need to be careful with resource management

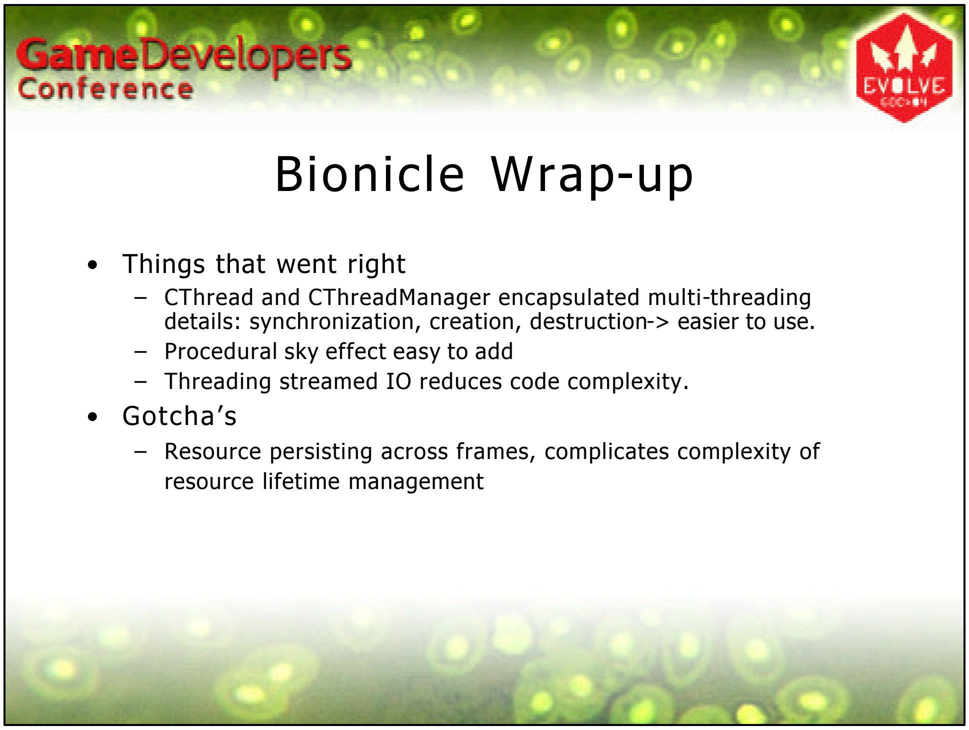
Before

After



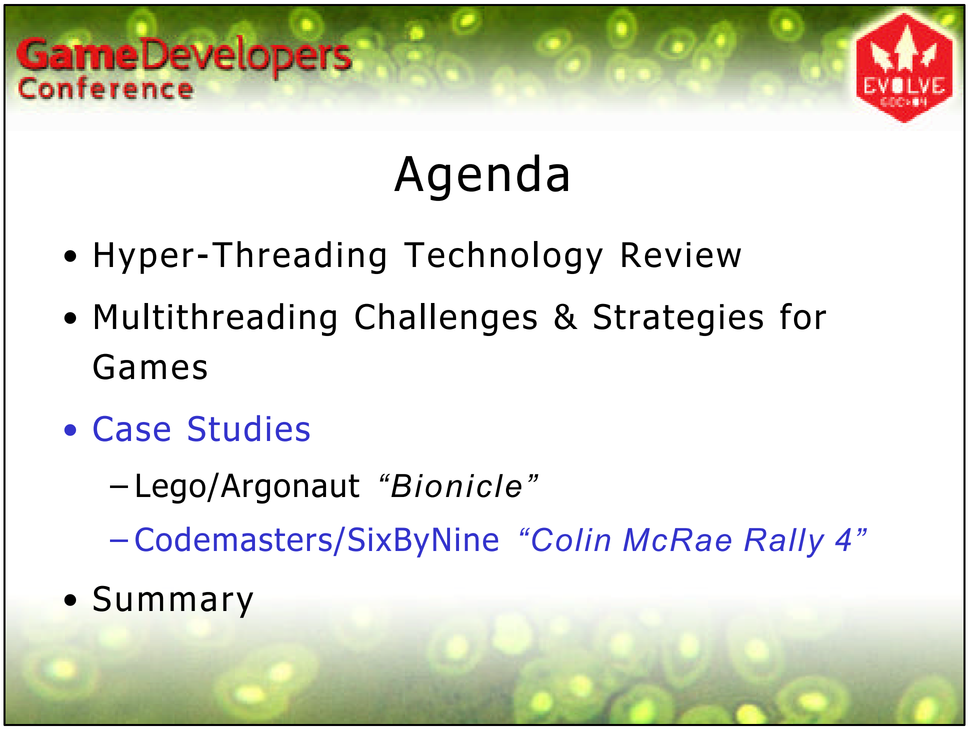
Background Resource Streaming

- File Read and Decompress in Second thread
  - Task Level Parallelism
- Pro:
  - Common code base across OS’s -> reduced code complexity and better bug repro.
  - Some additional performance gained by multi-threading blocked-IO
- Con:
  - Hard to abort File loading operations, impact on switching streams at will



Bionicle Wrap-up

- Things that went right
  - CThread and CThreadManager encapsulated multi-threading details: synchronization, creation, destruction-> easier to use.
  - Procedural sky effect easy to add
  - Threading streamed IO reduces code complexity.
- Gotcha’s
  - Resource persisting across frames, complicates complexity of resource lifetime management

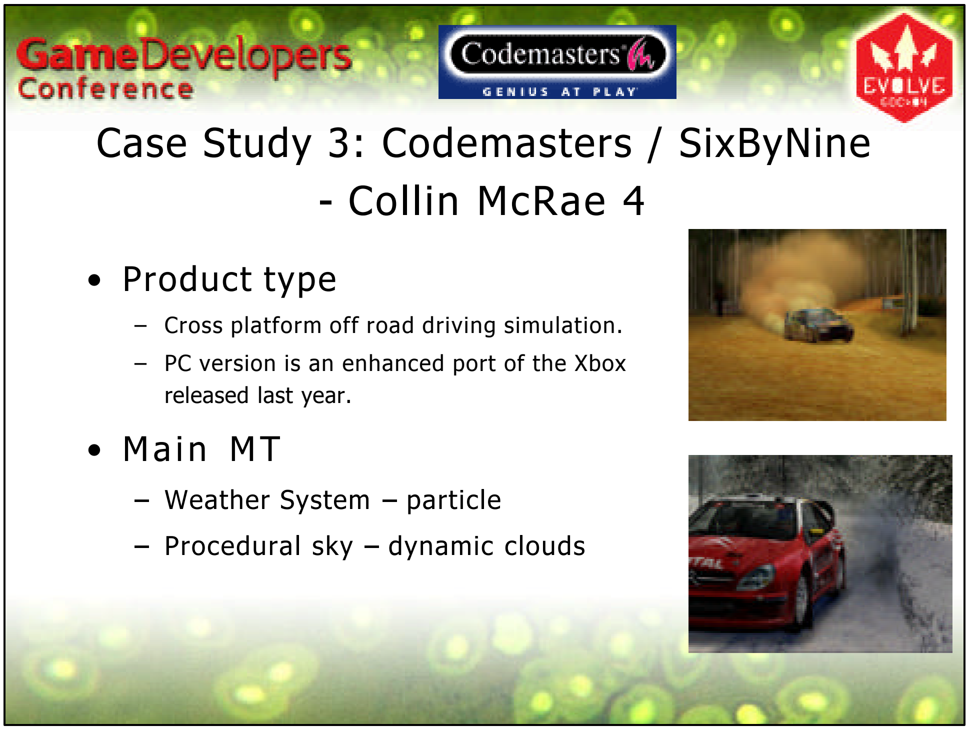


GameDevelopers Conference

EVOLVE GDC>04

Agenda

- Hyper-Threading Technology Review
- Multithreading Challenges & Strategies for Games
- Case Studies
  - Lego/Argonaut *“Bionicle”*
  - Codemasters/SixByNine *“Colin McRae Rally 4”*
- Summary



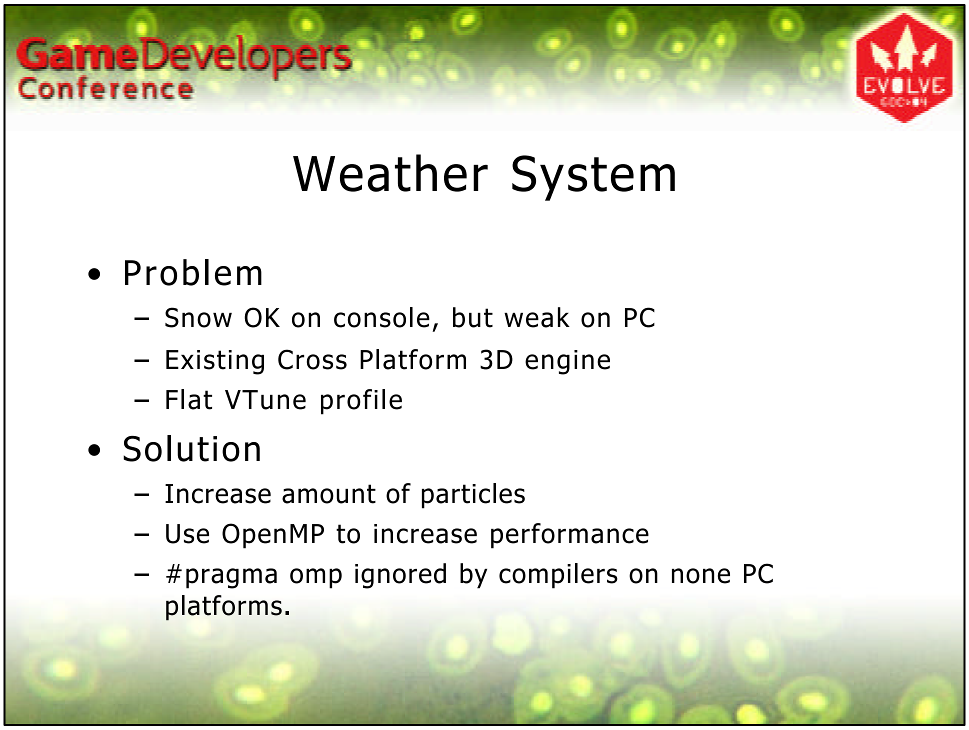
GameDevelopers Conference

Codemasters GENIUS AT PLAY

EVOLVE GDC>04

Case Study 3: Codemasters / SixByNine - Collin McRae 4

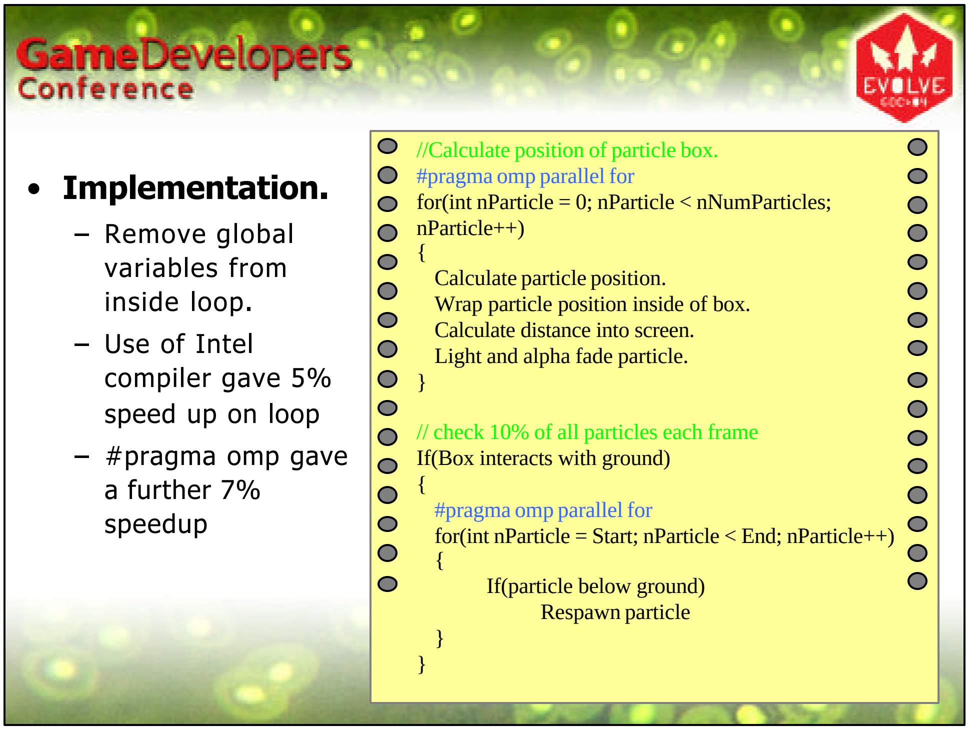
- Product type
  - Cross platform off road driving simulation.
  - PC version is an enhanced port of the Xbox released last year.
- Main MT
  - Weather System – particle
  - Procedural sky – dynamic clouds



GameDevelopers Conference

Weather System

- Problem
  - Snow OK on console, but weak on PC
  - Existing Cross Platform 3D engine
  - Flat VTune profile
- Solution
  - Increase amount of particles
  - Use OpenMP to increase performance
  - #pragma omp ignored by compilers on none PC platforms.

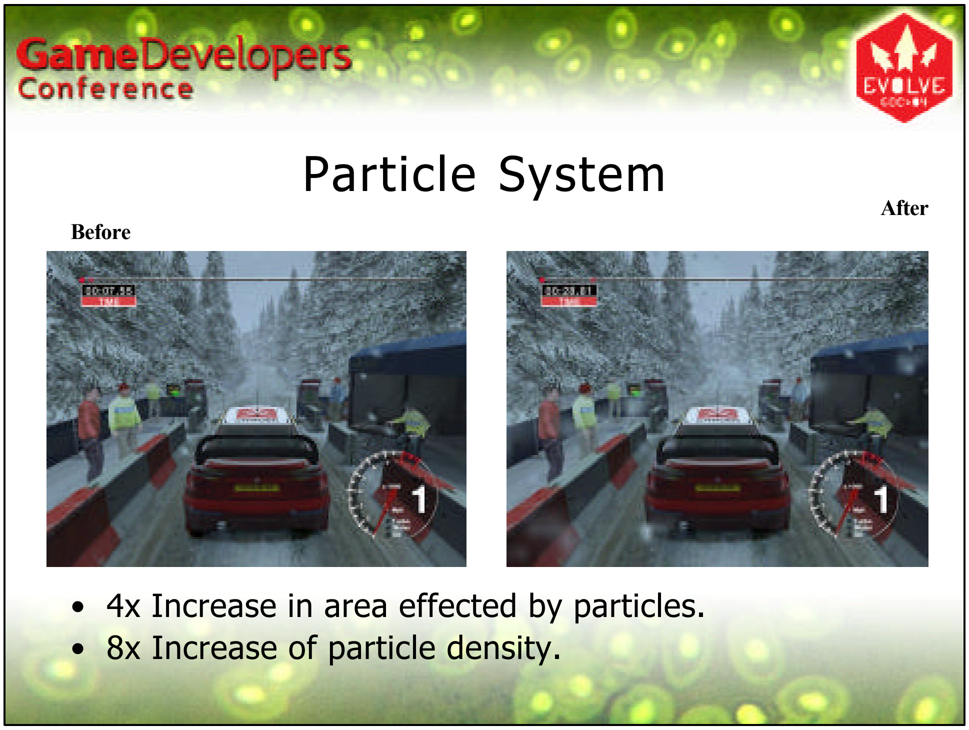


GameDevelopers Conference

- **Implementation.**
  - Remove global variables from inside loop.
  - Use of Intel compiler gave 5% speed up on loop
  - #pragma omp gave a further 7% speedup

```
//Calculate position of particle box.
#pragma omp parallel for
for(int nParticle = 0; nParticle < nNumParticles;
nParticle++)
{
   Calculate particle position.
   Wrap particle position inside of box.
   Calculate distance into screen.
   Light and alpha fade particle.
}

// check 10% of all particles each frame
If(Box interacts with ground)
{
   #pragma omp parallel for
   for(int nParticle = Start; nParticle < End; nParticle++)
   {
          If(particle below ground)
                 Respawn particle
   }
}
```

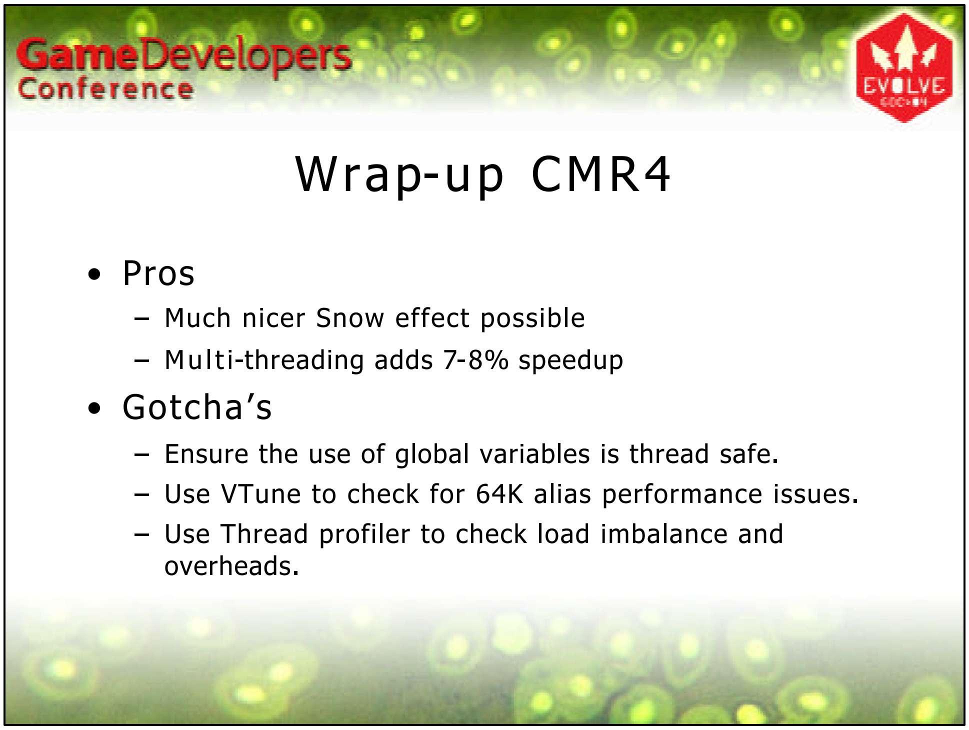


Particle System

Before

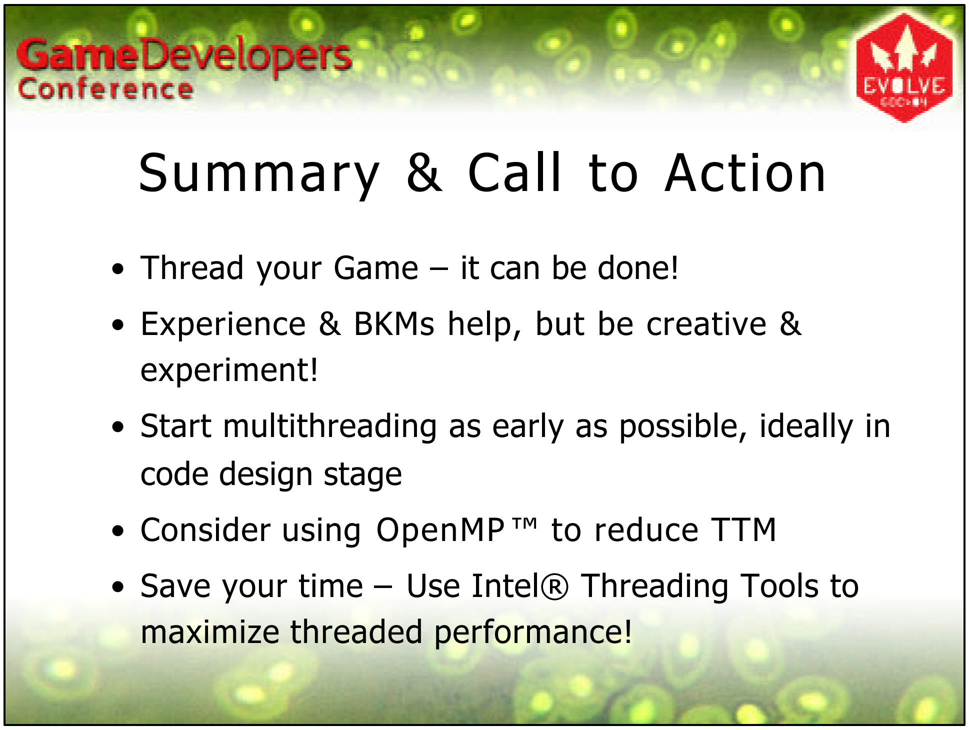
After

- 4x Increase in area effected by particles.
- 8x Increase of particle density.



Wrap-up CMR4

- Pros
  - Much nicer Snow effect possible
  - Multi-threading adds 7-8% speedup
- Gotcha's
  - Ensure the use of global variables is thread safe.
  - Use VTune to check for 64K alias performance issues.
  - Use Thread profiler to check load imbalance and overheads.


GameDevelopers
Conference
EVOLVE
GDC•04

Summary & Call to Action

- Thread your Game – it can be done!
- Experience & BKMs help, but be creative & experiment!
- Start multithreading as early as possible, ideally in code design stage
- Consider using OpenMP™ to reduce TTM
- Save your time – Use Intel® Threading Tools to maximize threaded performance!