

The Continuous World of Dungeon Siege

GDC 2003

Scott Bilas

Gas Powered Games



"Everything on fire, all the time"

Cell Phones?

Overview

- What is Continuous World?
- Concepts
- Terrain
- Game Objects
- World Streamer
- Odds and Ends

What is Continuous World? V1.0

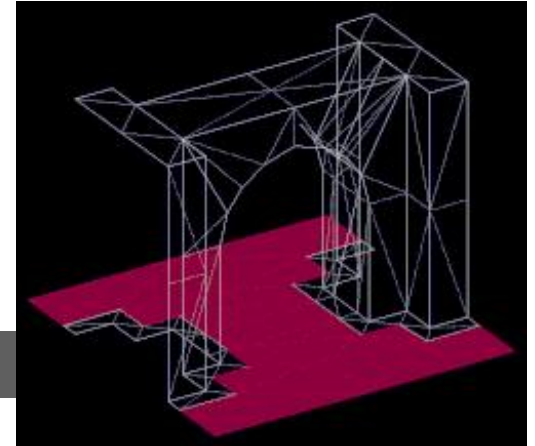
- Many games have similar sounding feature
 - Jak & Daxter, Drakan PS2, Spec Ops, flight sims...
- What does it mean to Dungeon Siege?
 - No loading screens except initial game load
 - Gameplay never stops from begin to end
 - Seamless indoor/outdoor transition
 - Extreme changes in environment
 - Constantly changing working set of resources
 - 8-Way client/server multiplayer (#@!!)

What is Continuous World? V2.0

- Realized new advantages during development
 - Fine-grained streaming avoids choke points
 - No arbitrary constraints in any direction
 - Extreme density and variety of content possible
- Scary things too
 - Teleportation across the map
 - Took over entire game despite our best efforts

(Fun) Demo!

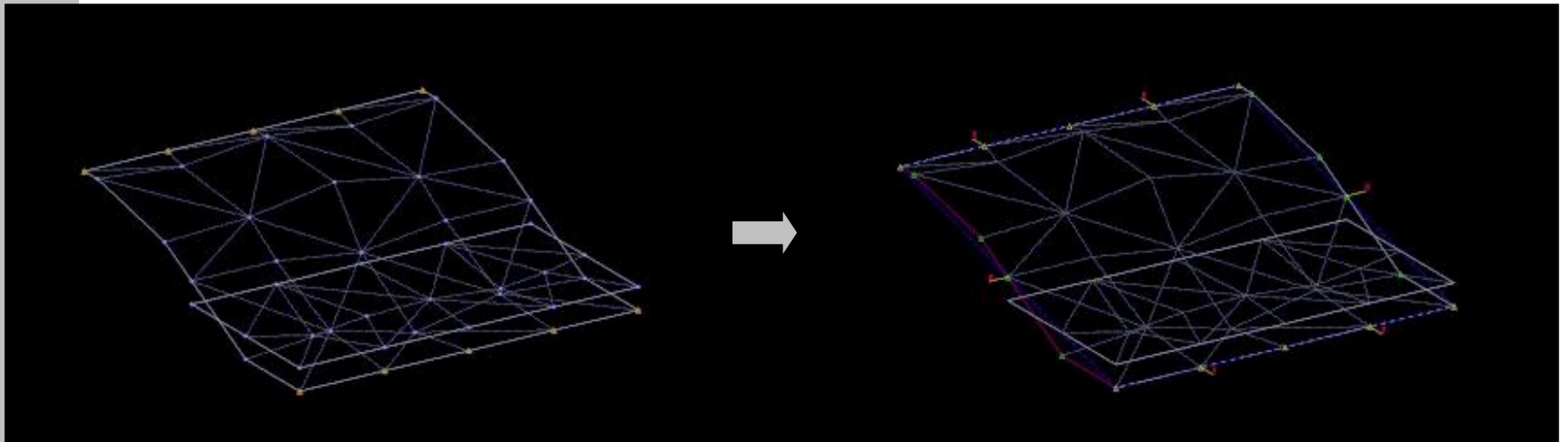
Concept: Siege Node



- Basic 3D terrain element (“tile”)
 - Drives entire game
- 3D artist makes mesh and exports to game
 - Mesh is completely arbitrary, no engine constraints on size, shape, lights, textures, connections...
 - Can mark polys as floor, water, etc.
- Map = instanced meshes placed into graph
 - Placement done with Siege Editor
 - Similar to Lego system “snap pieces together”

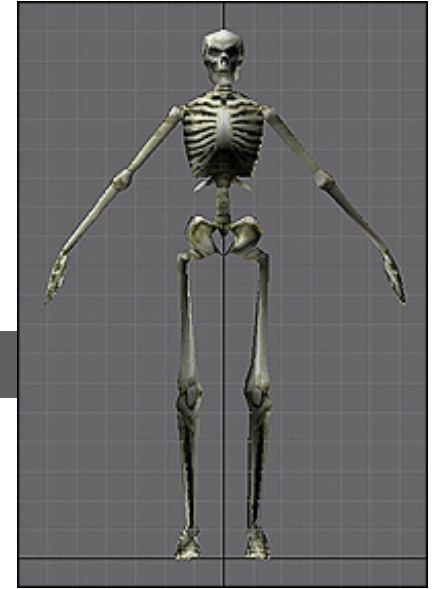
Concept: Siege Door

- Nodes are connected along doors
 - Legacy term from when they really were doors
- Artists choose verts for each door in Max tool



Concept: Game Object

- Represents all non-terrain and interactive logic content
- Similar: Entity, Actor, Object, Pawn, etc.
- 99% are based in Siege Nodes
 - Node = spatial owner (this is key)
- Result: terrain engine drives game objects
 - Nodes act as buckets for spatial sorting
 - Nodes used for relative queries (“who’s near me?”)



Concept: World “Frustum”

- Visualize as a box moving through the world
 - One owned by and centered on each party member
 - Intersect with world node graph to decide which nodes and Go's are kept in memory and active
 - Box dimensions configurable by code or content
 - Large superset of view frustum
 - Anything outside the box *does not exist*
- Term is misleading and inaccurate (sorry)

(Happy) Demo!

The Precision Issue

- Gigantic continuous world \neq numerical stability
- Increasing distance leads to quantized space
 - Eventually everything is “in the same place”
 - Increasing float precision will not solve
- Conclusion: axe the unified coordinate system
 - Segment the world
 - Periodically reset precision by switching coordinate systems
 - Now you can go in any direction forever, worry free!

The Precision Issue (cont.)

- Experimented for a while
- Ended up with variation on portal engine
 - Each chunk of geometry has its own “space”
 - Geometry (nodes) are linked together into terrain
- Evolved beyond FPU precision solution
 - Became primary method of subdividing space
 - Root of countless optimizations
- “There is no world space!”

Engine Mechanics

- 3D Position had to be augmented
 - SiegePos = node ID + x,y,z (relative to node origin)
 - Can represent a position anywhere in the map
- Rendering to avoid seams
 - Just render nodes on top of each other
 - “Stitching” to form continuous mesh not necessary

Engine Mechanics (cont.)

- Node graph = entire continuous world terrain
 - Each node has a unique ID
 - Linking done through doors
 - “Door” really = “transform” under the hood
- Engine is 3D, but up is always up
 - Nodes are rotated in flat space to hook together
 - Can think of engine as old fashioned 2D tiles
 - Permits significant optimizations (e.g. pathfinder)
 - Had to alter some design elements such as flying

Constructing Worlds

- Maps built using our Siege Editor tool
 - Choose a start node, the type of node you want to place near it, and flip through orientations
 - Drop objects into nodes and customize properties
 - Repeat hundreds of thousands of times!

Sarah Boulian is giving a talk on this in an hour
(go see it!!)

Constructing Worlds (cont.)

- Maps broken into regions
 - Editor not continuous; edit in chunks that fit in RAM
 - “Stitch” regions together, game sees as continuous
- Allows terrain that “bends space”
 - Convenient for designers (easier to make things fit)
 - Fading = interpenetrating terrain invisible to player
 - Goofy possibilities: infinite desert, moebius strips

(Fading) Demo!

“World Space”

- Ok, we actually *do* have world space
 - For one frame
 - Maybe longer, but don't count on it
 - Need world space for diff calcs, render tris, etc.
- Space is tracked by choosing a “target node”
 - This node *defines* space (its origin = world origin)
 - We just use the center of the current world frustum
 - Frustum moves with each party member
 - Party member crosses node: new coord system

“World Space” (cont.)

- Why change at node boundaries?
 - Good balance of testing vs. efficiency
 - Need to change as often as possible to avoid boundary conditions
- The Space Walk
 - New coordinate system = must rebuild space
 - Each node requires transform to target node space
 - Walk outward from target node, visit neighbors, accumulate transforms (similar to skeletal animation system)

War Story

The “Arrow Problem”



- Walking process
 - Find containing node for arrow to collide Go's
 - Relative coords requires starting node for ray trace
 - Ray trace has to walk outwards to max depth
 - Arrows can fly right through people!
- Arrows would break every couple weeks
 - Fading, scaling, attaching, spawning, collisions...
 - Had frequent problems with node-straddling systems (projectiles, particle system, etc.)



Evicting Nodes

- Frustum = cache management system
 - Anything inside is kept active in memory
 - Anything outside is thrown away or put on death row
- Algorithm
 - Walk outward from target node
 - Any nodes intersecting frustum box are loaded
 - All others are deleted, and contained Go's notified that they have “left the world”
 - Complicated by multiple frustums!

Multiple Frustums

- Originally implemented for party-split feature
 - Later required for multiplayer anyway
- Implementation
 - Multiple simultaneous coordinate systems
 - “Glomming” technology to determine winner
- Single player still needs it
 - One is considered “active” and its contents get time
 - Everything else, time is frozen (by design, but good for CPU also)

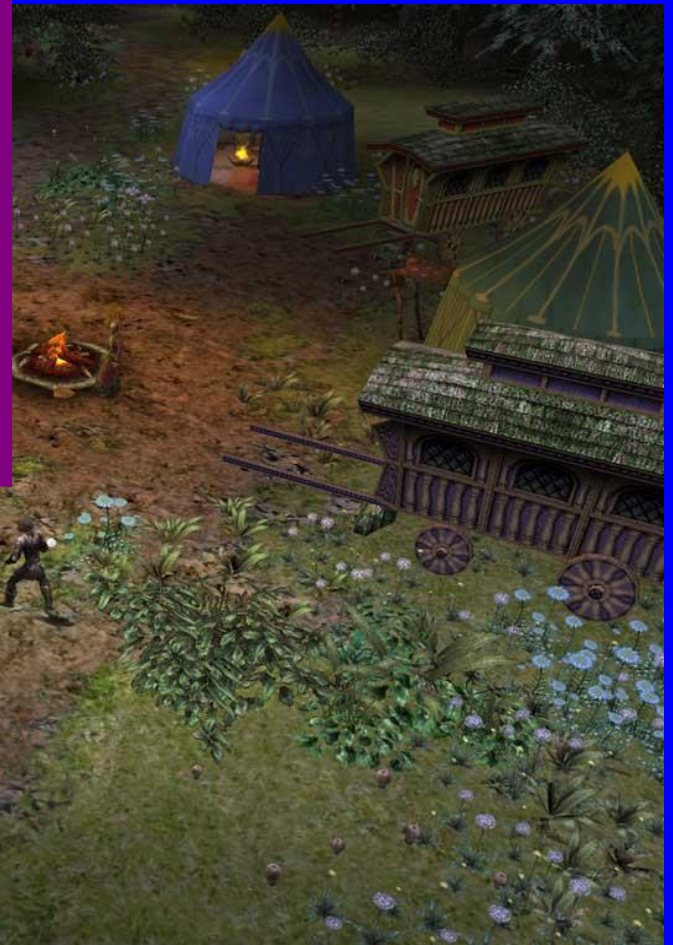
(Multi-Frustums) Demo!

Great World Detail

- World originally sparsely populated
- Artists started experimenting with density just to see how it would perform, what it would look like, and...









Evicting Game Objects

- With 60,000 Go's per map, very important
 - That can take up a lot of memory
 - Have to classify critical/non-critical
- Deciding what to throw out (not like nodes!)
 - Keep everything: machine runs out of memory
 - Keep nothing: lose critical quest states and “have I killed boss X and gotten loot Y” triggers
 - Want to try for “keep nothing” but err on the side of “keep everything”.

Eviction Strategies

- Fluff removal
 - lodfi
- Automatic expiration on leaving the world
- Scid retirement
 - (Scid = static content identifier)
 - Try to self destruct everything possible
- Data reduction
 - Scidbits
 - Model/texture purging

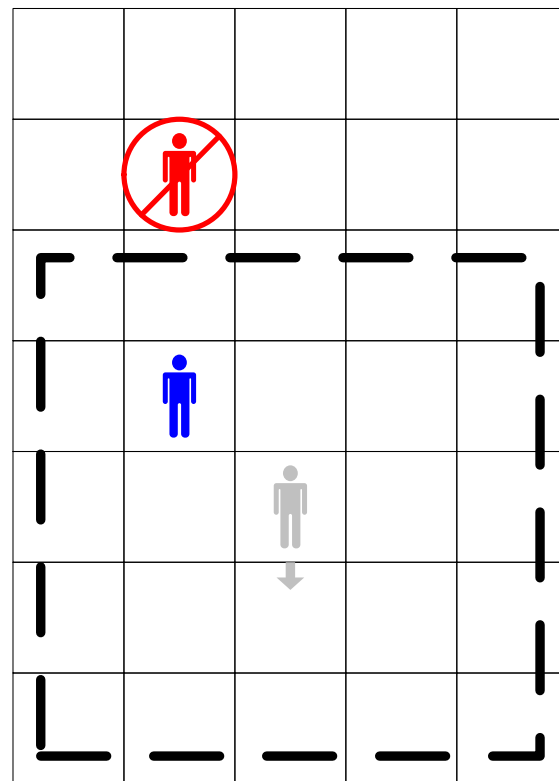
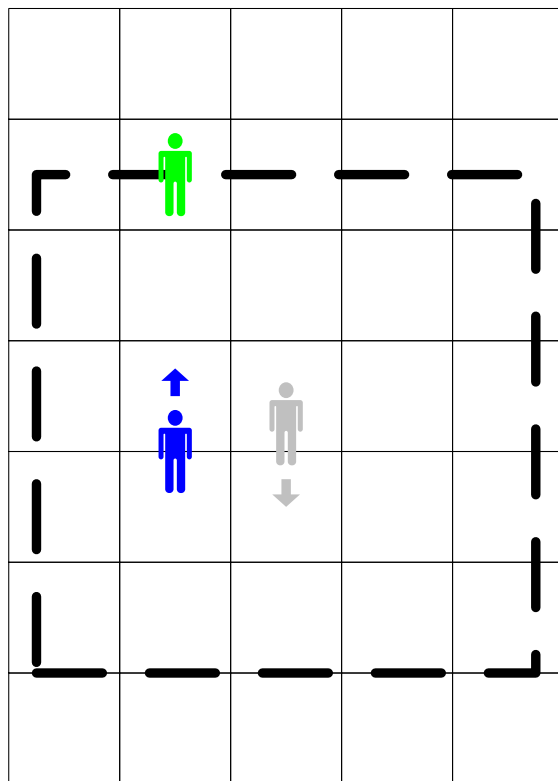
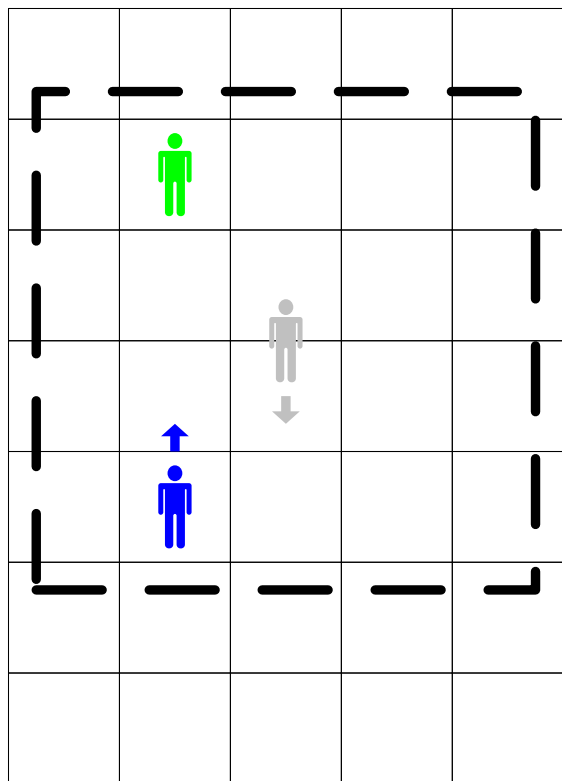
Continuous Logic

- Building interactive logic without levels is hard
 - No fixed places (like level transitions) to delete everything old and load everything new
 - No entry/exit points to hang scripted events
 - Long term game stability far more important
 - Difficult to affect objects not immediately nearby
- Not only is world continuous, but *logic* is too!
 - This took years for us to fully grok
 - Can't cover very much of it today!

Continuous Logic (cont.)

- Continuous world = constant change
 - 0.4% of the game's resources are in memory
 - Go's are constantly entering and leaving the world
 - Leaving is the hard one!
- Dependencies among Go's must be weak
 - Your referenced Go may leave world next frame
 - It may get deleted, too
 - A new one may get put in its place with the same ID (although this is unlikely)

Two men enter, one man leaves



Continuous Logic (cont.)

- Game must be very tolerant of failure
 - Especially at frustum boundaries
 - Added a number of self-healing features
 - Multiplayer complicated things a little

Multiplayer

- Each machine only knows about local frustum
 - Too expensive for bandwidth, CPU, otherwise
 - Server tells each client to create objects that are in its frustum, and deletes them outside (no expiration)
 - Frustum membership used to route RPC packets
- State delta transfer
 - Track “dirty” Go’s
 - Send delta packet with creation request
 - Transfer minimum visual data required for client

(Dirty Go's) Demo!

World Streamer Implementation

- Secondary thread, loads resources
 - Primary thread makes requests: nodes, textures, Go's
 - Textures use blank (or white) placeholder on load
 - Go's fade in when loaded
 - System bets on the objects being there

Streamer Problems

- CPU performance
 - Must be $< 20\%$ on second thread or player notices
 - Want to keep the load steady, not bursty
 - Original intention was DMA only, but zlib killed that
 - Kept load balanced by throttling work order filling
- Continuous performance
 - We are experts at thread contention (*not good!*)

Thread Contention

- Bad threading model for Go's
 - Much of Go load path on second thread
 - ...including parameterized content (gah!)
 - Most of game had to be thread protected
 - SmartHeap had to run serialized (5% perf loss!)
- Not fully solved
 - Still hitches due to lack of serious time spent on it
 - Maintenance of systems too difficult

Odds and Ends

- Teleporting hackery (H.U.B. system)
 - Became critical must-have feature
 - Engineering nearly drank the Kool-Aid
- Implementation
 - Elevator system and invisible nodes
 - Wacky node swapping
 - Frustum size and fog changes to smooth load
 - Complicated level designer wiring of objects
 - It works!! Ship it!!!

(Teleporter) Demo!

Pathological Cases

The Castle Ehb

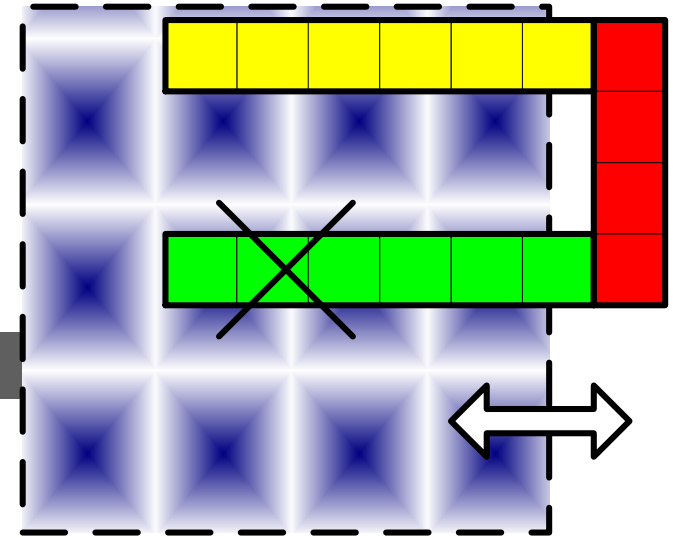


- “It’s slow”
 - Blamed on high poly for a long time (partially true)
 - Was at end of game so few played it
- Profiling reveals...
 - Implicit optimizations towards farmhouse nodes
 - Lighting expects small nodes (relatively few verts)
 - Game db’s expects few node occupants
- Lesson learned
 - Um, play the game... (not obvious!)



Pathological Cases

U-Shaped Terrain



- Direction of motion through world matters!
- Exposed extremely obscure bug
 - Occurred in a totally normal-looking area
 - Caused Edge-of-World Syndrome™
 - Complicated by multiple frustums, as usual ☺
- Lesson learned
 - Build test maps for all possible boundary conditions
 - Not as obvious as this may seem

Hardware Problems

- Setup issues collide with streaming data perf
 - Hard drive fragmentation
 - Heavy reliance on DMA transfers *enabled*
 - Other games just load slower – DS is paralyzed
- Exercise of most computer systems at once
 - CPU, video, HD, network, sound all constantly used
 - L33t overclocker troublemakers!
“Quake 3 runs fine”

Fun! Before and After



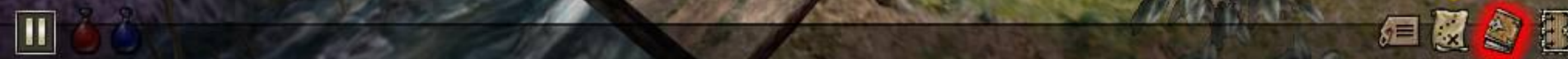




FIRESHOT

ENGAGE		[Icons]	
DEFEND		[Icons]	
TARGET CLOSEST		[Icons]	
[Icons]	[Icons]	[Icons]	[Icons]

8



13.4 FPS Speed 1.0 GameTime 00:17:14.4



Attack	Patrol
Run	Stop
Move	
Guard	

Fighting

Targeting

Movement



Attack

Defend

Character: The Hero
weapon: The Hero
life: 651.00
mana: 465.00
strength: 40
intelligence: 40
dexterity: 40



Further Reading

- GDC

- “Neverwinter Nights Client/Server Postmortem”, Mark Brockington, Scott Greig
- “Highly Detailed Continuous Worlds”, Stuart Denman
- “Building an Object System”, Alex Duran
- “Technology of Jak & Daxter”, Stephen White
- “A Data-Driven Game Object System”, Scott Bilas

Further Reading (cont.)

- Papers
 - “Postmortem: Gas Powered Games’ *Dungeon Siege*” by Bartosz Kijanka (on Gamasutra)
 - Be sure to read my paper in the Proceedings, it goes into details in many places this lecture did not

Done!

Suggested Comments:

This is the (best/worst) talk I've ever (been to/slept through)

<Insert name of speaker> changed my life forever.

I H4X0R YUO ALL!@!1!!

(He/she) didn't go into enough detail, and bored me to death.

(He/she) went into too much detail, and bored me to death.

Contact Info

Scott Bilas

<http://scottbilas.com>