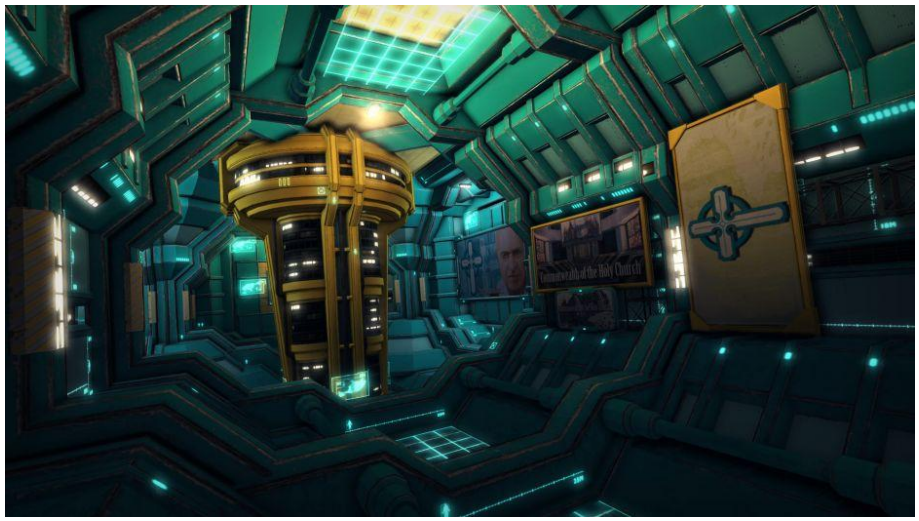
AAA game in XNA?

From the authors of Miner Wars 2081

Version: 27th March 2012

Marek Rosa, Keen Software House



www.minerwars.com

www.keenswh.com

Why XNA?

- XNA simplifies access to DirectX 9
- XNA is more intuitive than DirectX = less development time
- C# / .NET vs. C++
  - Faster compilation (seconds vs. minutes)
  - Code readability
  - Existing .NET classes and libraries
  - Unicode strings
  - Garbage collector
- 95% of game time is spent in GPU, since Miner Wars is heavily GPU bound, therefore it's usually irrelevant that C# code is slower than C++
- Probably easier porting and debugging to XBOX – *but we haven't tested it yet*

www.minerwars.com

Disadvantages of XNA

- We are bound to DX 9, e.g. can’t use texture atlas, tessellation, multi-thread rendering, etc. Not a problem right now since we have other priorities.
- Occasional bugs in XNA, e.g. device lost
- Can’t port to other platforms than Windows and XBOX

www.minerwars.com

Scene statistics

- Voxel asteroid: 2-10 mil triangles
- Space station: 5-10 mil triangles
- Sector/scene = 50x50x50 kilometers
  - Tens of voxel asteroids
  - Tens of space stations and space ships
  - Thousands of prefab modules
- We are aiming for:
  - 60 FPS, one frame has 16.6 milliseconds
  - About 500,000 triangles in frustum
  - About 1,000 objects in frustum

Voxels

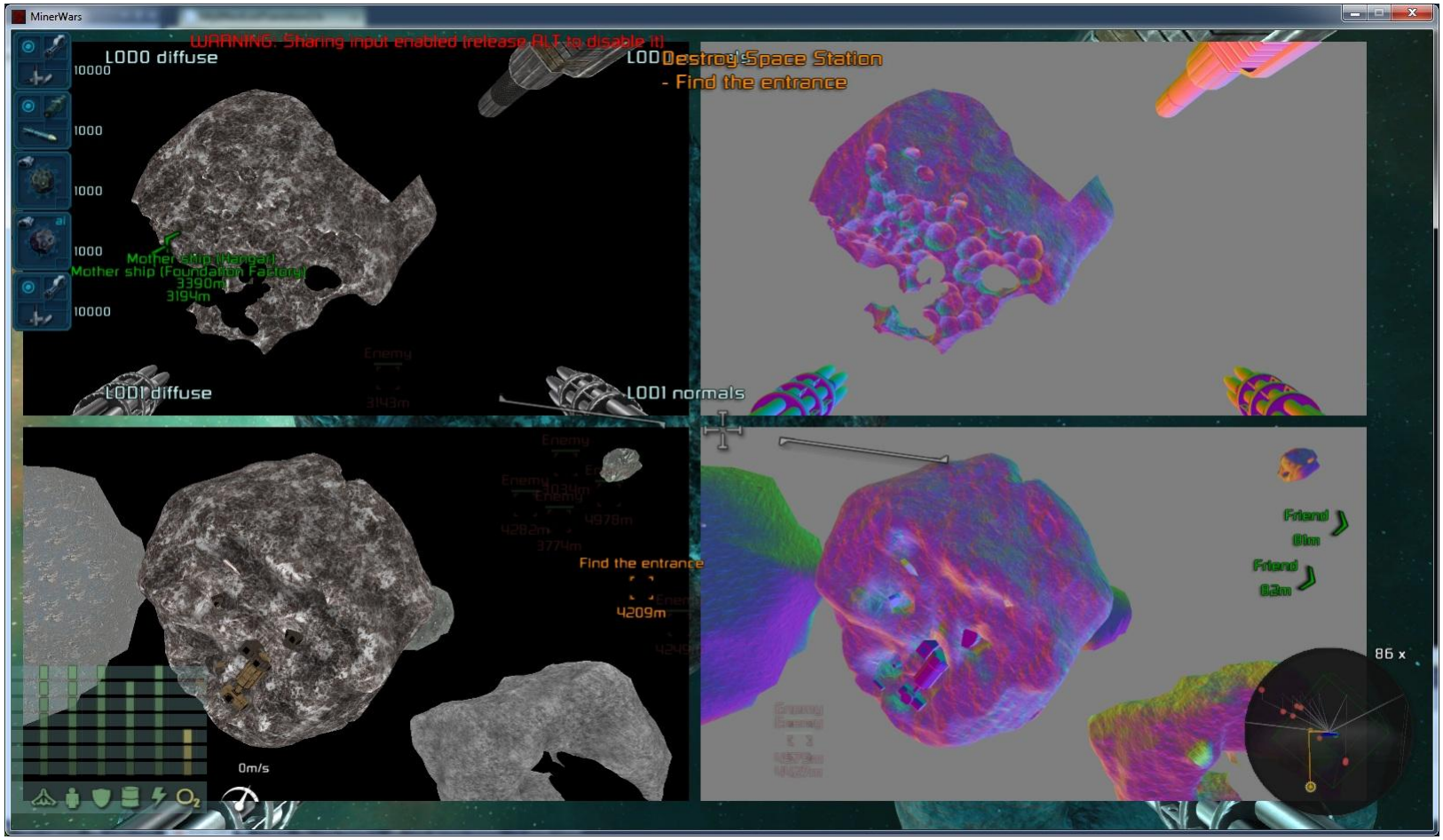
- Voxels are converted to triangles prior to rendering / collision-detection
- Cached in CPU (data cell, render cell)
- Cached in GPU vertex and index buffers (larger render cells with tens of thousands of triangles)
- Recalculated after a change (data/render cell invalidated after explosion or editor operation)

www.minerwars.com

Deferred Renderer

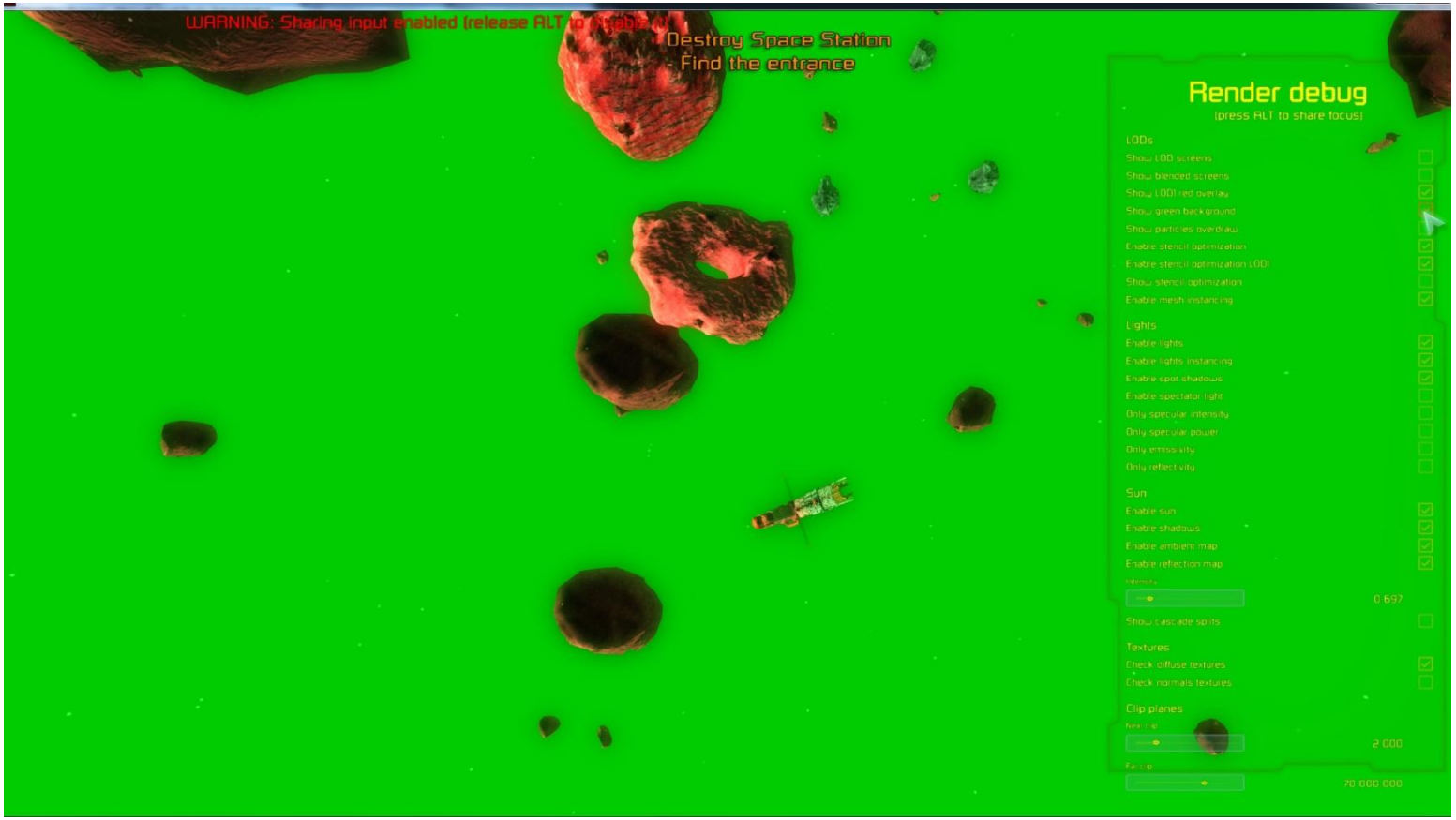
- Renders into 3 render targets (diffuse + specI, normal + specP, linear depth + emissivity)
- LOD0 & LOD1 – smooth LOD transition in image space (no popping or hard switching)
  - LOD0 = high-poly near objects, less than 2,000 meters distance
  - LOD1 = low-poly distant objects, more than 2,000 meters distance
    - LOD1 scene has usually 100-500x less triangles than LOD0
    - Some models don't have LOD1 (e.g. doors, small tunnels) – they fade-out to background
    - LOD1 for voxels is calculated by averaging voxel data cells
- Dynamic environment cube-map - re-rendered as you move, ambient map and reflection map represent the actual surrounding
- Sun light and shadows, point hemisphere pot lights
- SSAO
- FXAA
- HDR
- Transparent geometry (billboards, particle system – emitters, 2D animated parameters, generations)

Smooth LOD transition in image space



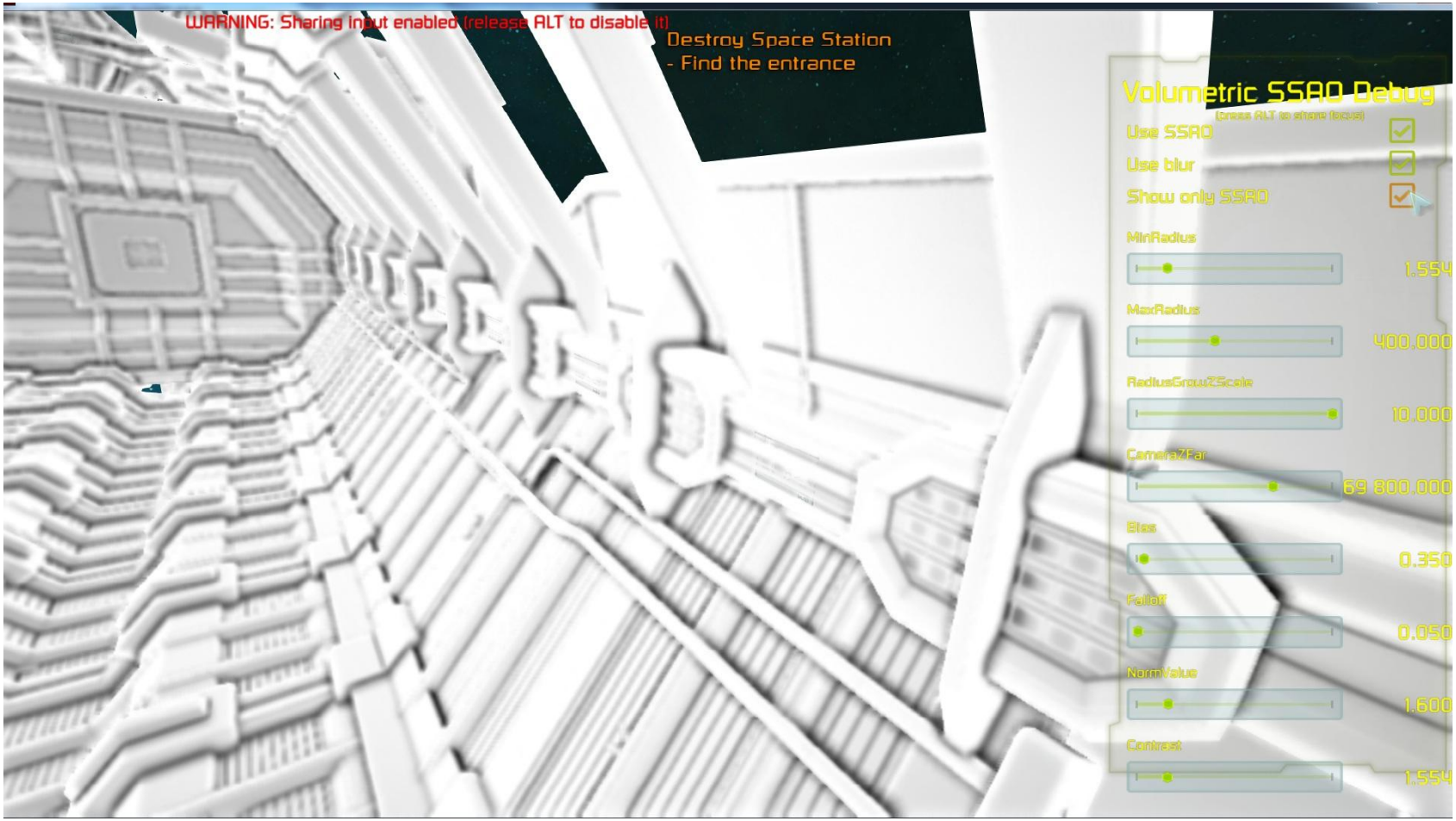
www.minerwars.com

Smooth LOD transition in image space



www.minerwars.com

Screen Space Ambient Occlusion - SSAO



www.minerwars.com

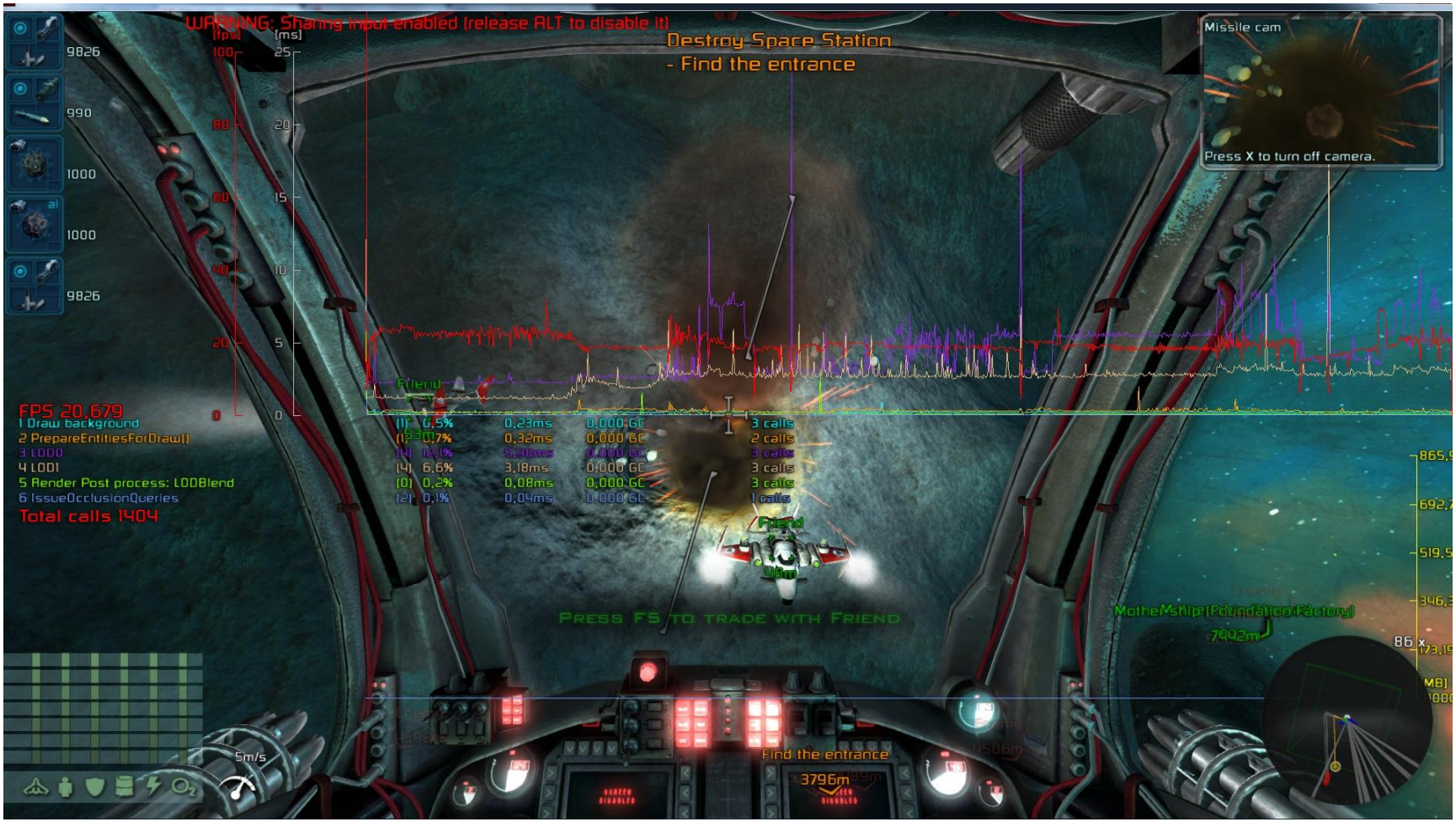
Physics & Simulation

- Own in-house physics engine
- Player and AI ships are represented as boxes (spheres work well too)
- Voxels can be represented as voxels or triangle meshes
- Spatial hierarchies
- Tunneling avoidance for high velocity objects by interpolating the position between steps (e.g. missiles or high speed ships)
- Integration step is 60x per second
- Area sensors

Optimizations

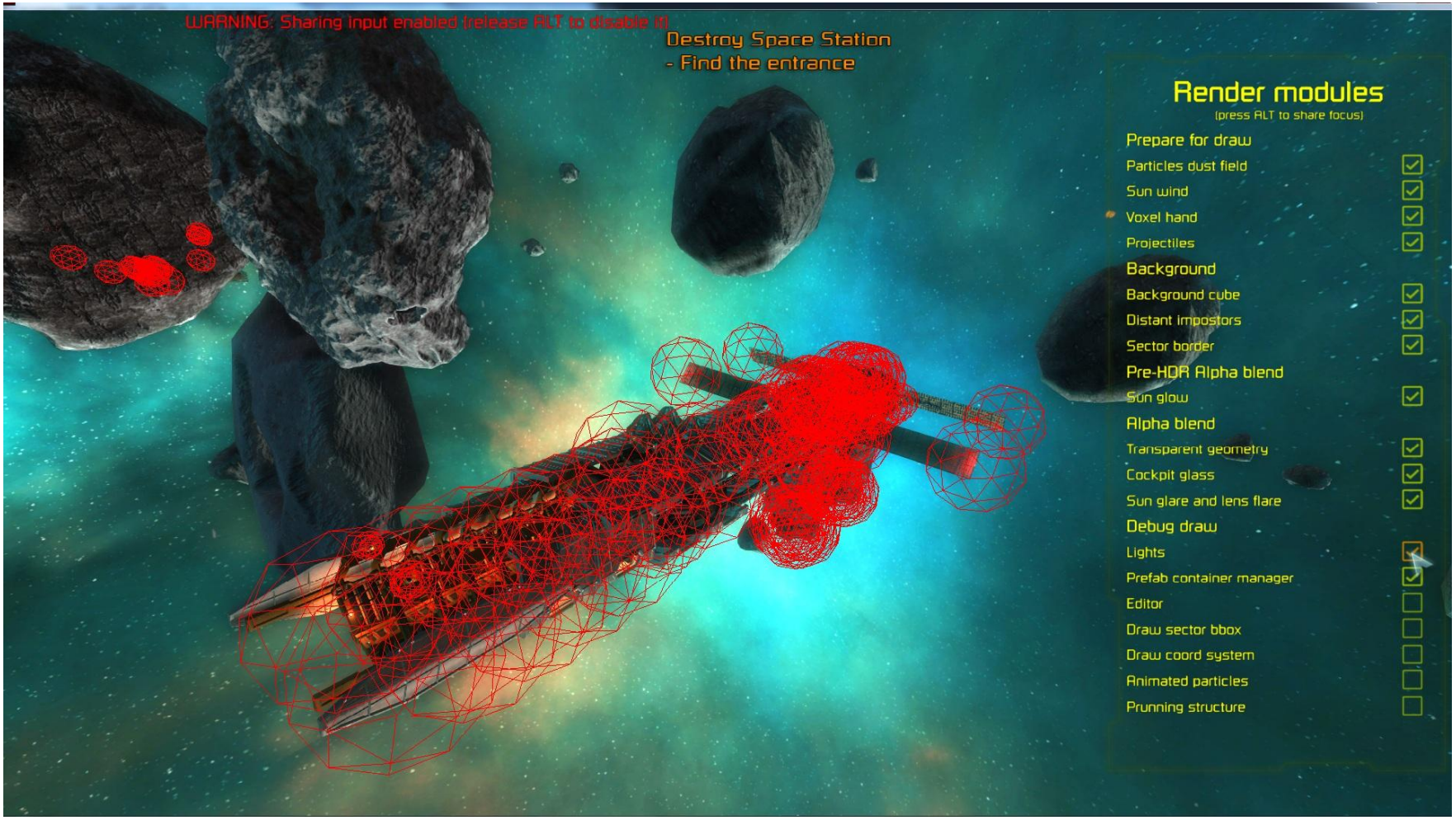
- We use our own in-game profiler, for performance & memory profiling
- Focus on “algorithm and idea” optimizations
- Hard-core low-level optimizations only in methods that are used frequently, e.g. collision detection calculations, pixel shaders, etc.
- Use pruning structures, AABB, octrees, multi-dimensional queues…
- Multi-threading for voxels and physics
  - We use Parallel Tasks
  - Consider task granularity, e.g. throwing a loop with 1000 simple iterations into parallel tasks won’t help. The task handling overhead is too big. Test it.
- Vector and matrix operations – use ref/out or write it yourself, it’s faster
  - E.g. **[a.x = b.x * c.x; a.y = b.y * c.y; a.z = b.z * c.z;]** is way more faster than **[a = b * c]**
- GPU instancing – use it for deferred lights and model rendering, about 5-10% speed improvement
- If possible, pass objects by reference, avoid structs (copying on stack is expensive)
- Don’t do object allocations during gameplay (calling “new” on a class). Use pre-allocated objects pools (particles, billboards, projectiles, debris objects, etc.)
  - Although, object allocations are OK during the game loading phase

In-game profiler



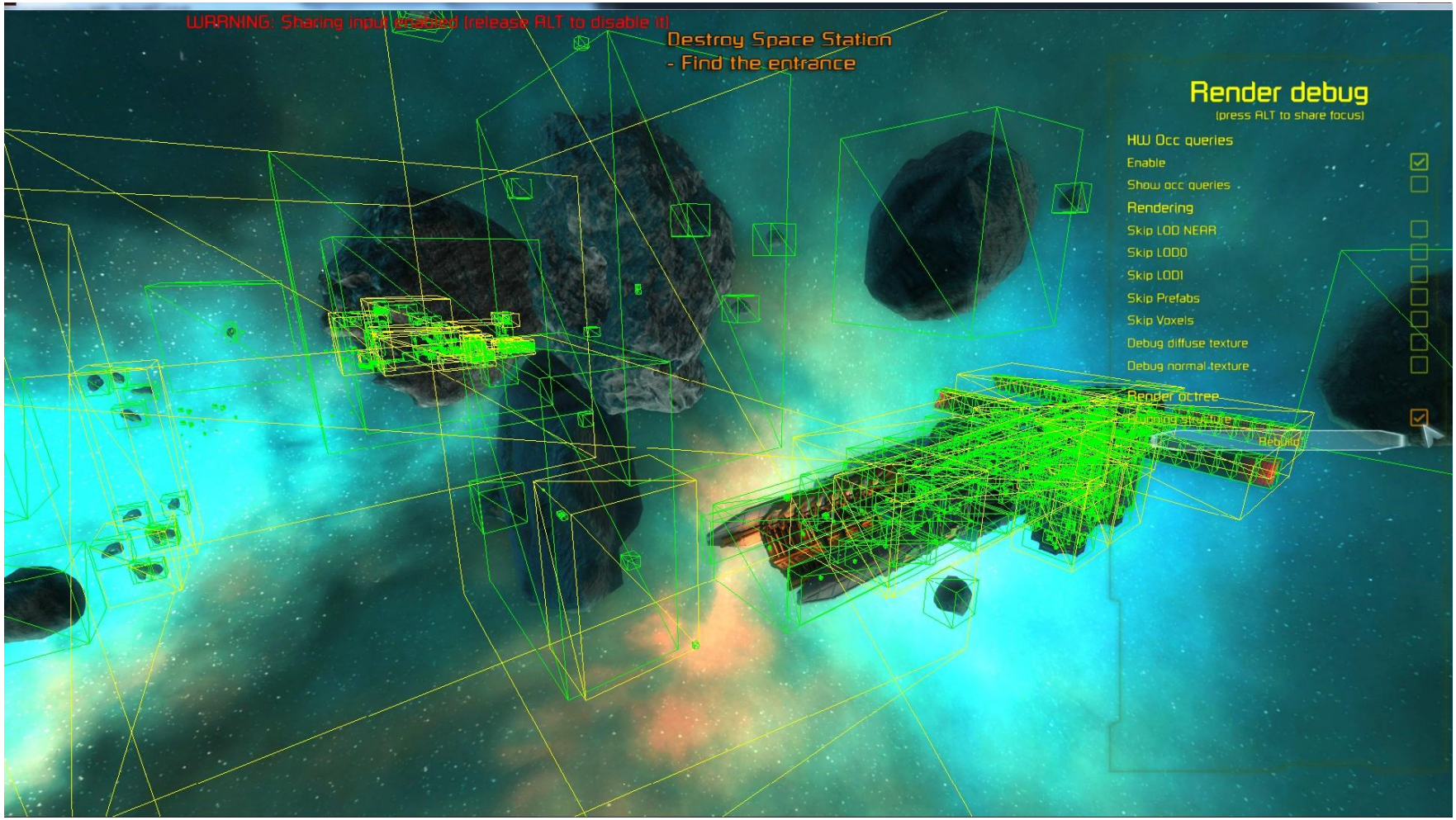
www.minerwars.com

Visualizing deferred lights



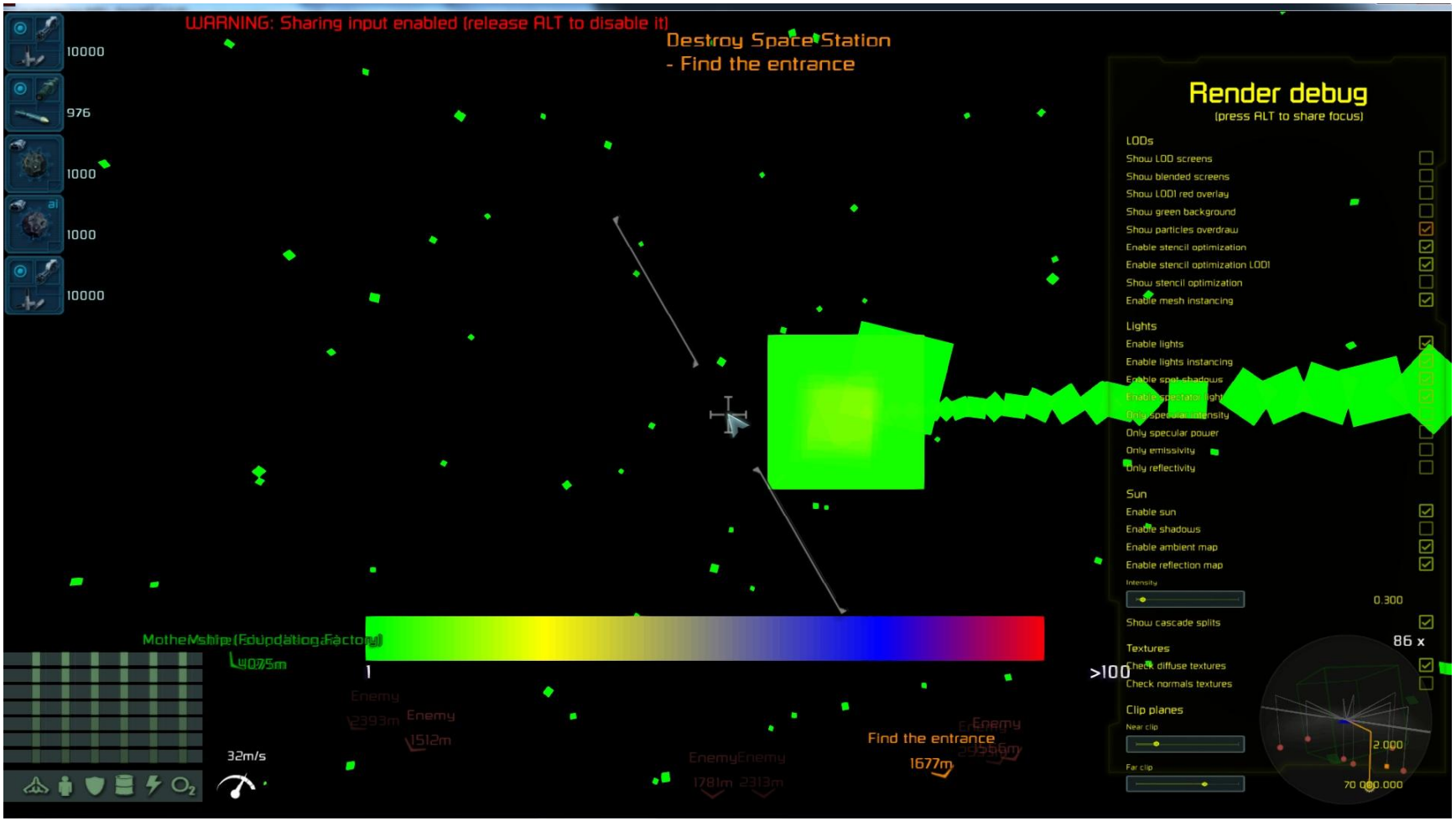
www.minerwars.com

Render octrees



www.minerwars.com

Visualizing particle overdraw



www.minerwars.com

Thank you

- We are still looking for programmers in Prague.
- Talented students too!

Marek Rosa, Keen Software House

www.minerwars.com

www.keenswh.com