Tinman 3D - EXECUTIVE SUMMARY

Matthias Englert | *Fulda (DE)* | *tinman3d.com* | *me@tinman3d.com* | *WEB*



Busy revolutionizing? Start evolving.

I've been writing software since the age of seven, starting with BASIC. What began as a hobby became my profession and never stopped being fun. My work ranges from commercial mobile games and industrial automation to real-time 3D graphics, algorithms, compiler technology and reusable software architectures.

I particularly enjoy building the foundations: understanding difficult problems, finding sound mathematical and technical solutions, questioning established assumptions, designing clear interfaces and implementing software that other developers can build upon.

Explore the Executive Summary
Ask about Matthias Englert's professional experience, Code-X, Tinman 3D SDK, and potential commercial collaboration.

AT A GLANCE

- **Code-X Workflow:** An independently developed, deterministic source transformation workflow. A defined, syntactically and semantically supported subset of C# is transformed into working C++, HLSL, GLSL and Java source code.
- **Tinman 3D SDK:** A modular software development kit for real-time 3D terrain visualization, geospatial data processing and spatial applications. Use the components you need rather than adopting an entire engine.
- **Selected engineering work:** From the long-lived Java ME client of TibiaME to industrial systems integration, multi-robot scheduling and routing for GroFi®, and published terrain-rendering research.
- **Approach:** Hands-on architecture, algorithms, implementation, testing, documentation and collaborative engineering. Deterministic tools and well-defined interfaces matter more to me than fashionable development slogans.

The Code-X Workflow and Tinman 3D SDK are distinct projects with a close technical relationship: significant parts of the SDK are written in Code-X compliant C#.

For technical details, follow the project links and documentation below. This summary is available as both HTML and PDF, generated from one AsciiDoc source. Naturally. ;-)

PROFESSIONAL BACKGROUND

I grew up programming CGA, EGA and VGA hardware, fascinated by real-time computer graphics. I still enjoy both the theory and the practical work of software engineering.

FOCUS

- Software architecture, algorithm design, framework and SDK development, compiler technology, GPU programming, cross-platform development and real-time 3D rendering.
- Collaborative engineering, mentoring, pair programming and code reviews, with a preference for clear responsibilities and maintainable implementations.
- Cooperation around the Code-X Workflow and Tinman 3D SDK, including SDK evaluation and licensing.

ENGINEERING APPROACH

I like exploring unconventional ideas, challenging assumptions and turning concepts into working software. My interest in AI follows the same practical approach: rather than redesigning established development workflows around AI agents, I prefer giving them access to well-structured environments, reliable tools and clear constraints.

WORK

- **Tinman 3D SDK** (2012 - today)
  Creator and business owner - responsible for research and development, technical implementation, product development and business operations
- **Freelance software development** (2003 - today)
  Started during my studies. Selected engagements across different industries:
  Airbus Helicopters (commercial research), CipSoft GmbH (mobile development, later freelance engagement), Daimler AG (production systems), DLR (manufacturing execution systems), Deutsche Lufthansa AG (intranet applications and reporting), EDAG Engineering GmbH (production systems), Flight Data Vision (3D applications), United Navigation GmbH (embedded systems).
- **BiteTheBytes UG (haftungsbeschränkt)**, Fulda (Nov 2007 - Nov 2011)
  Co-founder. Independently conceived, designed and implemented the experimental Cloddy terrain SDK prototype.
- **Games Academy GmbH**, Frankfurt (2010 - 2011)
  Lecturer. C++ programming, software design, algorithms and data structures, and classical game AI, using BattleBots as a teaching environment.
- **CipSoft GmbH**, Regensburg (Nov 2005 - Nov 2006, employed; later also freelance)
  Senior Software Developer (mobile games). Independently reimplemented the complete TibiaME MMORPG client from scratch in Java ME (J2ME), using the founders' original C++ client as a technical and functional reference. Rather than a straightforward port, the work involved designing and implementing a functional client under the severe memory, performance and

compatibility constraints of early mobile phones. The J2ME client was released commercially and remained in use for many years. I remain grateful for the trust, freedom and genuinely supportive working environment, and have great respect for the founders' well-deserved success.

SELECTED WORK

- **GroFi® MES (DLR, Mar 2018 - Sep 2022):** Designed and implemented the core architecture and major components of a modular industrial software platform and SDK for the GroFi® automated production environment, replacing an unsuitable legacy framework.
  - **Industrial integration:** Connected Siemens S7 PLCs over TCP/IP, robots through MCIS over Microsoft RPC, and a proprietary production system, using consistent APIs and abstraction layers.
  - **Architecture and tooling:** Built a modular Java3D-based visualization framework with rendering decoupled from application logic, and a model-driven PLC code generator using Program Structure Information (PSI) and source transformations.
  - **Multi-robot scheduling:** Developed A*-based scheduling with simulated robot trajectories and motion profiles from the Siemens Virtual NC Kernel (VNCK) to coordinate programs and avoid collisions.
  - **Multi-robot routing:** Implemented global A*-based routing for robots sharing a constrained rail network, modeling rail occupancy and robot ordering, including temporary repositioning to resolve blockages.
  - **Technical leadership:** Hands-on architecture and implementation, mentoring, pair programming, code reviews and project coordination. Primarily Java, with selected C# and C++ integrations.
- **Cloddy (diploma thesis, 2004 - 2005):** Designed and prototyped an efficient algorithm for visualizing complex distributed terrain landscapes, combining algorithmic research with practical real-time 3D implementation. Developed a benchmarking framework and evaluated performance; the framework also supported further benchmarking work by a fellow student. The thesis was titled *"Entwurf eines effizienten Algorithmus zur Visualisierung komplexer verteilter Landschaften und eine exemplarische Implementierung"*. The underlying CLOD idea began around 2000/2001 with sketches and experiments on paper. The diploma thesis subsequently provided the basis for EXIST-SEED start-up funding in 2007, followed by my experimental Cloddy SDK prototype and eventually the commercial Tinman 3D SDK from 2012 onward.

SELECTED INDEPENDENT PROJECTS

Early software development and technical exploration (age seven - 2003)

My father introduced me to programming when I was about seven. I soon continued learning independently, progressing from BASIC and Turbo Pascal to inline assembler, C++ and low-level PC programming. These were primarily personal projects driven by curiosity rather than formal assignments. Much of the original early code was lost; the examples below are reconstructed from memory.

- **Games and development tools:** Built 2D games ranging from an early CGA spaceship game to multiplayer tilemap combat games. Wrote a pixel-art and animation editor whose assets were saved to files and loaded by the games, later adding sprite rotation. Completed an EGA (640 × 350, 16 colors) adaptation of the board game *Das verrückte Labyrinth*, with menus, up to four players, reachability display and automated shortest-path movement. The exact pathfinding algorithm is no longer known.
- **Mathematics and visualization:** Implemented Mandelbrot and bifurcation-diagram visualizations, and a mathematics program with its own text user interface and custom overflow-avoidance arithmetic for large binomial coefficients. Later generated triangle meshes for mathematical surfaces of revolution and displayed them interactively with my own 3D renderer.
- **Hardware control and practical software:** Developed a CGA-based application to display terrain cross-sections and, as far as I recall, calculate areas for my father's civil-engineering work. We also programmed a Fischertechnik robot arm through a digital/analog interface and its Turbo Pascal SDK, controlling motors and counting optical position-sensor pulses. A positioning error taught us to keep counting pulses even after switching off the motors, because the mechanism continued moving.
- **Low-level graphics and reverse engineering:** Worked directly with CGA, EGA, VGA and VESA graphics, including Mode X, scrolling, page flipping and register-level effects. Reverse-engineered DOOM graphics and its palette-based lighting. Investigated the proprietary data of *M.A.X. (Mechanized Assault & Exploration)* with a hex editor and custom visualizations, then built a graphical level editor that wrote playable custom maps back into the original game's map slots. Also experimented with PC-speaker, AdLib and Gravis UltraSound audio interfaces.
- **3D graphics and terrain:** Built a 16-bit VESA software triangle rasterizer with Gouraud shading, dynamic brightness-based lighting, perspective-correct texture mapping, alpha blending and depth-sorted rendering (Painter's Algorithm). Experimented with voxel-based terrain and later with 3dfx/Glide and Direct3D hardware rendering. A terrain prototype evolved from a uniform grid to several levels of resolution, with additional triangles closing gaps at the boundaries; visible bends remained. This was an independent, simpler approach, not an implementation of the later published Geometry Clipmaps method.

Toward the beginning of the 2000s, limitations of then-current per-triangle CLOD methods led me to explore an alternative, semantically organized and index-free terrain data structure. That work began as sketches and experiments on paper, before its later implementation and evaluation in the Cloddy diploma thesis described above.

Tiny Engine and Mothership!Attack (after graduation)

I developed **Tiny Engine**, a reusable C# framework for 2D arcade games, and used it to create **Mothership!Attack**, a space-themed arcade shooter originally built as a hands-on way to learn C#. The linked LinkedIn post includes a gameplay video with audio.

Tiny provides managed game objects with explicit update and lifecycle rules, entities and behaviors, sprite and particle rendering, collision detection and spatial queries, resource management, input and audio abstractions, and interfaces that separate engine code from platform-specific implementations. Mothership!Attack builds on these foundations with enemy ship classes, attack waves and formations, different weapons and effects, shields, heat management,

upgrades, menus, high scores and keyboard/mouse or gamepad controls.

Both codebases survive. Adapting them to the Code-X workflow and using them as educational examples are future plans, not claims of a completed port.

BattleBots - game AI teaching environment (late 2000s / Games Academy)

I developed **BattleBots** as a practical environment for teaching C++, algorithms and data structures, classical game AI and client/server interaction. It is a turn-based combat game on a tilemap: the game server receives commands from human or AI-controlled players and processes movement, attacks, repairs and other effects according to defined rules.

The accompanying *AI in Games* course material develops a modular AI design around **Knowledge, Analysis, Planning and Execution**. It covers a game-world model; line of sight using Bresenham's algorithm; spatial analysis of passability, visibility, danger and distance; pathfinding with different evaluation criteria; and goal-oriented state machines for selecting tactical behaviors.

The intended student exercise was to implement competing AI clients. That complete classroom scope proved too ambitious, but the game and teaching material are preserved. Bringing the codebase into Code-X and revisiting AI-client experiments are future plans.

PUBLICATIONS

- Modeling Asteroid (101955) Bennu as a Real-time Terrain (SIGGRAPH 2021 Talks, Aug 6, 2021)
  Demonstrates a hybrid representation of asteroid Bennu's complex surface using digital elevation models and static and dynamic displacement, and adapts tri-planar material mapping (TRIMAP) to planetary curvature to avoid texturing artifacts.
- Using Mesh Shaders for Continuous Level-of-Detail Terrain Rendering (SIGGRAPH 2020 Talks, Aug 17, 2020)
  Examines strengths and limitations of continuous level-of-detail terrain rendering compared with hybrid and implicit approaches, and shows how NVIDIA Turing mesh shaders and inline ray tracing can address its limitations.

HONORS & AWARDS

- **European Innovative Games Awards (EIGA), 2010:** The experimental **Cloddy** terrain project won category "T"; the prize was endowed with EUR 10,000. Associated with BiteTheBytes UG (haftungsbeschränkt).
- **EXIST-SEED start-up funding, 2007:** The **Cloddy** project received approximately EUR 70,000 in business start-up funding under the German Federal Ministry of Education and Research's EXIST-SEED program. Associated with the University of Applied Sciences Fulda.

EDUCATION

- University of Applied Sciences Fulda (2000 – 2005)
  Diplom-Informatiker / Graduate Computer Scientist, Telecommunications
  Final grade: 1.3 (sehr gut / very good)

CODE-X WORKFLOW

> I transform a defined subset of C# into source code for supported target languages. My transformation process is algorithmic and deterministic, not AI-based. Clear language rules, explicit restrictions and reproducible output are features, not limitations.



In a nutshell...

The Code-X Workflow uses the Code-X Processor to read Code-X compliant C# code and generate functional source code for supported targets, including C, HLSL, GLSL and Java. The supported C# subset and target-specific restrictions are explicit. Code-X originated during development of the experimental Cloddy SDK, before the launch of Tinman 3D in 2012. Having experienced aging codebases in BASIC, Pascal, assembler and especially C, I wanted to keep a long-lived SDK implementation actively maintainable. C# offered value types and generics; C remained a desired native target, but I no longer wanted to hand-maintain the equivalent implementation in C. This pragmatic requirement led to the Code-X workflow. Code-X combines a dedicated framework, processor and development tools with parsing, semantic validation, target-language and API restrictions, and development-time compatibility inspections. Native-code regions allow platform-specific implementations where the shared C# subset is not the right fit, rather than restricting every implementation to the lowest common denominator. Typically, the workflow is integrated into established development and build pipelines.

In other words...

Developers work within an explicit supported language subset, encouraging clear design and well-defined boundaries from the start (which may hurt initially, but pays off later). Teams can optimize their skill set and knowledge transfer: use common C# skills for essential algorithms and logic and rare skills only for the tricky parts.



Separating Code-X compliant C# from platform-specific code encourages explicit interfaces and responsibilities. It can reduce unnecessary coupling and make later changes easier.



Unit tests for the authoritative C# implementation can be reused as part of a consistent validation strategy. Generated targets still require appropriate integration, platform and runtime testing. The benefit is maintaining shared logic in one place rather than manually porting each change.



Code-X includes semantic restrictions and checks, including rules concerning resource ownership and memory consistency. These can help detect or prevent specific error classes; they are not a general guarantee against all vulnerabilities.

Example 1. Business / Game Logic

Write **business rules / game mechanics** in **simple C# code**, without settling for a particular third-party framework or engine. This **reduces complexity**, which **frees brain capacity** to go into design, features and correctness. Use Code-X to generate **functional code** for supported targets in your ecosystem. Additional languages and platforms are possible directions for further development, not a claim of current support.

Example 2. Software Library

Significant parts of the TINMAN 3D SDK are developed using the Code-X Workflow. Their authoritative C# code is maintained by developers; supported CPU and GPU target code is **generated automatically**. Fixes, changes and features only need to be implemented in the C# code. From there, changes are propagated to supported targets through the generation pipeline, without manually rewriting the shared logic.

TINMAN 3D SDK

tinman3d.com | *features* | *demo* | *getting started* | *online manual* | *licencing*

> I am a modular software development kit for real-time 3D terrain visualization and spatial applications. Rather than requiring an entire engine, I provide a structured toolbox of components: choose the parts that fit your application.



Independently developed since 2012, the SDK combines reusable libraries, development tools, application components, and platform and graphics API abstraction layers with clearly defined interfaces. The focus is on reusable foundations rather than application-specific solutions. Its terrain-meshing roots trace back through the experimental Cloddy SDK prototype to my earlier CLOD research; the present SDK is a much broader product, not merely a renamed thesis project.

Disclaimer / To whom it may concern...

Should you be looking for a full-blown **3D engine** or a dedicated **GIS application**, the SDK will most likely **not** be a match for you (but you could still use it to write plug-ins, though).

On the other hand, if you are looking for a **flexible toolset designed for software developers** building real-time 3D and GIS applications, you might want to **give it a try**!



Evaluation is free and requires no signup!



Technical questions and integration requirements can be discussed directly, with an emphasis on understanding the actual engineering problem.

Geodata Handling

- Process large geospatial datasets locally without requiring cloud services
- Work with multi-resolution data using streaming and spatial data structures

Data Import / Export

- Import elevation models, imagery, 3D models, 2D shapes and map data
- Combine and modify data at runtime or offline
- Export processed data for use in other applications

Real-time Rendering

- Render large-scale terrains efficiently with modern GPU features
- Tune rendering for frame rate and resource usage
- Support multi-view rendering scenarios

Offline Meshing

- Build continuous-level-of-detail terrain meshes
- Use CLOD meshes for visibility checks, collision avoidance, physics simulation, path finding and more

Cross-platform

- Designed for portability through structured platform abstraction layers
- Provides integrations for supported operating systems and libraries
- Platform support depends on the available backend and integration

Cross-language

- Shares algorithms and selected APIs across supported target languages through Code-X
- Uses explicit target-specific restrictions
- Additional language backends can be developed separately

COLLABORATION & CONTACT

I enjoy working with people who care about sound engineering, whether that means solving a difficult algorithmic problem, building reusable infrastructure, integrating existing systems or exploring a new application for Code-X or Tinman 3D.

A concrete technical question is always a good starting point. Contact: me@tinman3d.com | LinkedIn | tinman3d.com.

Busy revolutionizing? Start evolving.