Chapter 5

Qengine (3D)

Quansistor Field Simulator in a Volumetric Substrate

ProFCrank

Abstract

This paper introduces *Qengine (3D)*, a three-dimensional extension of the Quansistor Field Simulator. Qengine (3D) generalizes the planar quansistor substrate into a volumetric lattice while preserving the core semantics of operator-based computation, strict locality, causal update rules, and invariant-driven auditability.

As in the two-dimensional formulation, each lattice cell hosts a quansistor—an operator-level unit that exposes only locally computable invariant observables within a bounded sensing radius. Mobile cells are defined as finite, composite clusters of quansistors embedded in the volumetric substrate. Their motion, rotation, and deformation emerge from interaction with local invariant fields rather than from global control, explicit state transitions, or signalling channels.

The paper formalizes the volumetric substrate model, mobile cell geometry, local update rules, causal scheduling, and distributed execution under region-based partitioning. Deterministic replay and auditability are achieved through invariant commitments rather than exhaustive execution traces, enabling reproducible and verifiable execution on message-driven substrates such as the Internet Computer.

Qengine (3D) demonstrates that quansistor-based, operator-first computation scales to higher-dimensional substrates without sacrificing causal integrity or audit-native verification. The model serves as an experimental platform for exploring QVM semantics, emergent volumetric dynamics, and distributed execution under increased topological complexity.

1 Motivation for a Three-Dimensional Extension

Qengine (3D) extends the two-dimensional quansistor field simulator into a volumetric setting. The purpose of this extension is not to increase realism for its own sake, but to explore how operator-based, CA-like dynamics behave when spatial structure, locality, and interaction topology are enriched by an additional dimension.

The core principles of Qengine remain unchanged. The three-dimensional model preserves locality, causality, determinism, and invariant-based auditability, while enabling new classes of emergent phenomena that cannot arise in purely planar substrates.

1.1 Why Move Beyond Two Dimensions

The two-dimensional formulation of Qengine provides conceptual clarity and visual accessibility. However, many structural and topological features of distributed systems are inherently three-dimensional.

Extending the substrate to three dimensions enables:

- volumetric neighborhood structures,
- richer connectivity graphs,
- topological configurations unavailable in two dimensions.

The 3D extension therefore broadens the expressive capacity of the model without altering its foundational semantics.

1.2 Dimensionality as a Modeling Parameter

In Qengine, dimensionality is treated as a modeling parameter rather than a physical assertion. The transition from a planar grid to a volumetric lattice does not imply physical space, but rather an expanded interaction topology.

All causal constraints continue to be enforced through bounded sensing radii and local update rules.

1.3 Preservation of Core Semantics

Qengine (3D) preserves all core semantic properties established in the 2D model:

- quansistors remain operator-based units, not state holders,
- information is accessed only through invariant observables,
- updates are local and causally ordered,
- execution is deterministic and replayable.

No new communication channels are introduced by the additional dimension.

1.4 Motivation from Emergent Dynamics

Three-dimensional substrates support classes of emergent behavior that cannot exist in two dimensions, including:

- filamentary and volumetric structures,
- layered and shell-like configurations,
- three-dimensional attractors and defects.

Studying these phenomena provides insight into how operator-based locality scales with topological complexity.

1.5 Mobile Cells in Volumetric Space

Mobile cells embedded in a three-dimensional substrate gain additional degrees of freedom:

- orientation and rotation in three axes,
- volumetric deformation in elastic modes,
- more complex migration paths.

Despite this increased freedom, mobile cells remain constrained by local invariant gradients and causal update rules.

1.6 Relevance to Distributed Execution

From an execution perspective, three-dimensional partitioning aligns naturally with region-based decomposition on message-driven substrates.

Volumetric regions:

- generalize planar partitioning,
- increase boundary surface complexity,
- stress-test causal messaging and audit mechanisms.

This makes Qengine (3D) a valuable testbed for evaluating QVM-style execution semantics under increased structural complexity.

1.7 What This Extension Does Not Claim

Qengine (3D) does not claim to model physical spacetime, quantum fields, or biological organisms. The three-dimensional substrate is an abstract computational construct.

Analogies to physical systems are illustrative and should not be interpreted as physical correspondence.

1.8 Scope of This Paper

This paper focuses on defining the three-dimensional quansistor substrate and highlighting differences relative to the 2D model. Where semantics remain unchanged, the presentation refers implicitly to the 2D formulation.

Detailed implementation strategies and performance considerations are outside the scope of this work.

1.9 Roadmap

The next chapter defines the volumetric quansistor substrate, including lattice structure, sensing radii, and invariant fields in three dimensions. Subsequent chapters adapt mobile cell definitions, update rules, and distributed execution semantics to the 3D setting.

2 Volumetric Quansistor Substrate (3D Grid, Radius, Invariants)

This chapter defines the three-dimensional quansistor substrate underlying Qengine (3D). The volumetric substrate generalizes the planar grid introduced in the two-dimensional model while preserving identical semantic constraints on locality, causality, and invariant-based observation.

2.1 Three-Dimensional Discrete Grid

The substrate is defined as a three-dimensional discrete lattice

$$\mathcal{G}_3 \subset \mathbb{Z}^3,$$

where each lattice coordinate $(x, y, z) \in \mathcal{G}_3$ hosts exactly one quansistor.

The lattice provides:

- explicit volumetric locality,
- well-defined neighborhoods,
- a natural basis for spatial partitioning.

As in the 2D model, the lattice is an abstract computational construct and does not assert any physical interpretation of space.

2.2 One Quansistor per Lattice Cell

Each lattice cell contains a single quansistor $Q_{x,y,z}$. A quansistor remains an operator-level entity characterized by:

- a local operator or operator family,
- a bounded sensing radius,
- a set of locally computable invariant observables.

No explicit state variables are stored at lattice cells. All observable information is derived from operator structure.

2.3 Volumetric Sensing Radius

Each quansistor is associated with a sensing radius $R \geq 0$ defining a volumetric neighborhood:

$$\mathcal{N}_R(x, y, z) = \{(x', y', z') \in \mathcal{G}_3 \mid \|(x', y', z') - (x, y, z)\| \leq R\}.$$

The sensing radius:

- bounds information accessibility,
- enforces causal locality,
- may vary across the substrate.

Observation within the sensing radius does not imply direct control or modification of remote operators.

2.4 Neighborhood Geometry

In three dimensions, neighborhoods form volumetric regions rather than planar disks. This increases:

- the number of interacting neighbors,
- the diversity of interaction topologies,

- the smoothness of invariant gradients.

Despite increased connectivity, all interactions remain bounded and local.

2.5 Invariant Fields in Three Dimensions

Each quansistor computes invariant quantities derived from its local operator and the operators within its sensing radius. These invariant values define observable fields over the volumetric substrate.

Typical invariant fields include:

- stability or coherence measures,
- spectral gap or dispersion indicators,
- potential- or energy-like quantities,
- phase- or gauge-like parameters.

Invariant definitions are unchanged from the 2D model, differing only in spatial aggregation.

2.6 Invariants as Structural Observables

Invariant values are structural observables, not stored state. They may be recomputed as needed and need not persist beyond their evaluation window.

This ensures that:

- information remains representation-independent,
- no symbolic state propagation occurs,
- auditability relies on invariant reproduction.

2.7 Local Update Scope

Updates applied to a quansistor at (x, y, z) may affect only:

- its local operator parameters,
- invariant values derived from its neighborhood,
- subsequent evolution reachable through causal propagation.

No mechanism exists for instantaneous volumetric updates.

2.8 Substrate as a Volumetric Operator Field

The full lattice of quansistors defines a volumetric operator field. Spatial structure, interaction, and propagation arise from:

- lattice topology,
- sensing radii,
- invariant-based coupling.

The substrate itself remains passive, serving only to constrain and mediate local interactions.

2.9 Preparation for Mobile Cells in 3D

Invariant fields defined on the volumetric substrate provide the environmental context for mobile cells operating in three dimensions. Mobile cells respond to local invariant gradients and structural features without accessing global information.

The next chapter defines mobile cells in the 3D setting and examines how their geometry, coherence, and motion generalize beyond the planar case.

3 Mobile Cells in Volumetric Space

This chapter generalizes the concept of mobile cells from the planar formulation to a three-dimensional quansistor substrate. Mobile cells remain composite objects embedded in the operator field, but gain additional geometric and topological degrees of freedom afforded by volumetric space.

All semantic constraints established in the two-dimensional model are preserved.

3.1 Definition of a Volumetric Mobile Cell

A mobile cell in three dimensions is defined as a finite set of quansistors

$$\mathcal{C}_3 = \{Q_1, Q_2, \ldots, Q_N\},$$

together with:

- an internal coupling graph embedded in $\mathbb{Z}^3$,
- a coherence rule maintaining volumetric integrity,
- a sensing radius for interaction with the substrate,
- a motion functional defined on local invariant observables.

As in the 2D case, mobile cells are entities operating *on* the substrate, not modifications of the substrate itself.

3.2 Volumetric Cell Geometry

In three dimensions, mobile cells occupy finite volumes rather than planar regions. Typical geometries include:

- compact clusters (spherical or polyhedral),
- shell-like or hollow configurations,
- elongated or filamentary structures.

Geometry is a design parameter and does not alter the interface between the cell and the substrate.

3.3 Orientation and Rotational Degrees of Freedom

Volumetric space introduces full three-axis orientation and rotation. Mobile cells may:

- rotate about arbitrary axes,

- align with invariant gradients,
- exhibit tumbling or precessional motion.

Orientation is determined implicitly by local invariant structure and internal coupling rules, not by external control signals.

3.4 Rigid and Elastic Modes in 3D

As in the planar model, mobile cells may operate in different internal regimes:

Rigid mode. Relative distances and angles between constituent quansistors are fixed. The cell behaves as a rigid volumetric body undergoing translation and rotation.

Elastic mode. Relative positions are maintained via soft constraints. The cell may deform, compress, or stretch in response to invariant gradients while preserving overall coherence.

Both modes share identical interaction semantics with the substrate.

3.5 Interaction with Volumetric Invariant Fields

Each mobile cell samples invariant observables within a volumetric sensing radius. From these samples, the cell constructs local summaries such as:

- gradient vectors,
- directional contrasts,
- curvature-like indicators.

These summaries inform motion and deformation without accessing global information.

3.6 Motion in Three Dimensions

Motion in 3D arises as an emergent drift driven by a motion functional:

$$\mathcal{M}_3 : \{\text{local volumetric invariants}\} \to \Delta\text{configuration}.$$

The functional may induce:

- translational movement,
- rotational alignment,
- volumetric deformation.

No explicit velocity, force, or acceleration variables are required.

3.7 CA-like Behavior in Volumetric Space

Mobile cells in Qengine (3D) exhibit CA-like behavior in the sense that:

- evolution depends on local neighborhoods,
- rules are homogeneous across space,

- global structure emerges from repeated local interaction.

Unlike classical cellular automata, behavior arises from operator interaction with invariant fields rather than from state transitions.

3.8 Cell–Cell Interaction via the Substrate

Multiple mobile cells interact indirectly through the volumetric substrate:

- by perturbing invariant fields,
- by creating volumetric gradients sensed by others,
- by competing for stable regions.

No direct cell-to-cell signalling or shared state is assumed.

3.9 Coherence, Transformation, and Dissolution

A mobile cell persists as long as its coherence conditions are satisfied. In three dimensions, loss of coherence may lead to:

- fragmentation into multiple volumetric cells,
- collapse into a lower-dimensional structure,
- absorption into the substrate.

Such events are governed entirely by local invariant thresholds and internal rules.

3.10 Role in Qengine (3D)

Volumetric mobile cells serve as probes of three-dimensional operator field dynamics. Their trajectories, transformations, and interactions provide insight into how quansistor-based computation scales with dimensional complexity.

They remain abstract entities intended for experimentation, not physical or biological modeling.

3.11 Transition

Having defined mobile cells in volumetric space, the next chapter formalizes local update rules and causal scheduling for Qengine (3D), highlighting which aspects remain unchanged and which are stressed by three-dimensional partitioning.

4 Local Update Rules and Causal Scheduling in 3D

This chapter adapts the local update and causal scheduling framework of Qengine to a three-dimensional substrate. While volumetric space increases neighborhood complexity and boundary surface area, the semantic structure of updates remains unchanged.

The purpose of this chapter is to clarify which aspects of the update mechanism are dimension-independent and which aspects are stressed by three-dimensional execution.

4.1 Discrete Time and Update Phases

As in the two-dimensional model, time is represented as a sequence of discrete steps indexed by $t \in \mathbb{N}$. Each step consists of three causally ordered phases:

1. substrate update,
2. mobile cell update,
3. commit phase.

This structure is preserved verbatim in three dimensions.

4.2 Locality in Volumetric Neighborhoods

All update rules are strictly local. An update applied at lattice position (x, y, z) may depend only on:

- the local quansistor $Q_{x,y,z}$,
- invariant observables within its volumetric sensing radius,
- locally intersecting mobile cells.

No update may reference global information or distant regions outside its causal neighborhood.

4.3 Substrate Update Phase in 3D

During the substrate update phase:

- local invariant fields are recomputed over volumetric neighborhoods,
- operator parameters evolve according to local rules,
- mobile cell configurations remain unchanged.

The increased number of neighbors affects computational cost but not semantic behavior.

4.4 Mobile Cell Update Phase in 3D

In the mobile cell update phase, each volumetric cell:

- samples invariant observables within its sensing radius,
- evaluates its motion functional in three dimensions,
- computes translational, rotational, or deformational updates.

Cell updates are proposed independently and do not directly modify substrate operators.

4.5 Commit Phase and Volumetric Consistency

In the commit phase:

- proposed cell movements and deformations are applied,
- any local substrate effects induced by cell presence are committed,
- conflicts are resolved using deterministic, local rules.

Volumetric conflicts, such as overlapping cell regions, are resolved without global coordination.

4.6 Causal Ordering in Three Dimensions

Causal ordering is enforced through the same principles as in two dimensions:

- updates depend only on past or present local information,
- information propagates through successive local interactions,
- no update depends on spacelike-separated events.

The additional dimension increases the number of causal interfaces but does not introduce new signalling pathways.

4.7 Asynchronous Interpretation

The three-phase update scheme admits asynchronous execution in three dimensions. Volumetric regions may advance independently as long as causal dependencies at region boundaries are respected.

This property is essential for distributed execution on message-driven substrates with volumetric partitioning.

4.8 Determinism and Replay in 3D

Determinism is preserved under the same conditions as in the planar model. Given identical initial configurations, update rules, and message ordering, the three-dimensional evolution is fully reproducible.

Replay relies on invariant reproduction rather than procedural trace logging.

4.9 No-Signalling Guarantees

The increased connectivity of three-dimensional neighborhoods does not weaken no-signalling guarantees. All interaction remains bounded by sensing radii and causal ordering.

Volumetric proximity does not imply instantaneous influence.

4.10 Stress Points Introduced by 3D Execution

While semantics remain unchanged, three dimensions introduce practical stress points:

- larger neighborhood sizes,
- increased boundary surface area between regions,
- higher messaging volume under partitioned execution.

These stress points motivate careful partitioning strategies but do not alter the conceptual model.

4.11 Transition

With local update rules and causal scheduling established for volumetric space, the next chapter addresses execution of Qengine (3D) on an ICP-like substrate, focusing on region partitioning, messaging, and audit preservation.

5 Distributed Execution and Partitioning in 3D

This chapter describes how Qengine (3D) can be executed on a distributed, message-driven substrate under volumetric partitioning. The objective is to preserve locality, causality, determinism, and auditability while scaling execution across independent regions.

The execution model generalizes the planar partitioning strategy to three dimensions without altering core semantics.

5.1 Motivation for Volumetric Partitioning

The volumetric quansistor substrate naturally decomposes into spatial regions. Partitioning the substrate:

- enforces execution-level locality,
- enables parallel evolution of independent regions,
- avoids centralized coordination.

In three dimensions, partitioning also exposes the cost of increased boundary surface area, making it a valuable stress test for causal messaging semantics.

5.2 Volumetric Region Decomposition

The lattice $\mathcal{G}_3$ is decomposed into disjoint volumetric regions

$$\mathcal{G}_3 = \bigcup_i \mathcal{R}_i^{(3)},$$

where each region $\mathcal{R}_i^{(3)}$ is a contiguous three-dimensional block of lattice cells.

Each region:

- hosts a subset of substrate quansistors,
- manages mobile cells intersecting its volume,
- executes local update rules autonomously.

Region shapes and sizes are design parameters subject to locality constraints.

5.3 Mapping Regions to Execution Units

Each volumetric region is mapped to an independent execution unit. In an ICP-like environment, this corresponds to assigning each region to a canister or equivalent isolated runtime.

Execution units:

- maintain local operator and invariant data,
- perform substrate and mobile cell update phases,
- communicate only through defined boundary interfaces.

No execution unit has access to global substrate state.

5.4 Boundary Surfaces and Interfaces

In three dimensions, region boundaries are surfaces rather than lines. Each boundary interface exposes:

- invariant summaries for boundary lattice cells,
- mobile cell entry and exit events,
- causal step identifiers.

Internal operator details remain encapsulated within regions.

5.5 Message-Based Interaction

All inter-region interaction occurs via messages. Messages may convey:

- boundary invariant values,
- notifications of mobile cell migration,
- causal synchronization markers.

Messages are immutable and processed deterministically.

5.6 Causal Messaging Semantics

Each message is tagged with:

- originating region identifier,
- originating time step.

A region processes a message only if it originates from its causal past. This ensures:

- absence of future-state dependence,
- no instantaneous coordination across volumetric space,
- preservation of no-signalling guarantees.

5.7 Mobile Cell Migration Across Volumetric Regions

When a mobile cell crosses a region boundary:

- ownership is transferred to the destination region,
- the cell's internal configuration is serialized into a migration message,
- exclusive ownership is enforced.

Migration respects causal ordering and does not introduce duplication or race conditions.

5.8 Asynchronous Progress and Load Imbalance

Regions may advance asynchronously subject to causal constraints. Variations in local activity may lead to load imbalance, particularly in regions with:

- high mobile cell density,
- large sensing radii,
- complex invariant computation.

Such imbalance affects performance but not correctness.

5.9 Determinism Under Distributed Execution

Distributed execution preserves determinism provided that:

- initial configurations are identical,
- update rules are identical,
- message ordering is deterministic.

Given these conditions, volumetric execution yields identical invariant outcomes across replays.

5.10 Auditability Across Regions

Each region may periodically emit invariant commitments summarizing its local state. Global auditability is achieved by composing regional commitments without requiring global state aggregation.

This approach scales naturally to three-dimensional partitioning.

5.11 Stress Points Specific to 3D Execution

Three-dimensional partitioning introduces practical stress points:

- increased boundary surface area,
- higher message volume,
- more frequent mobile cell migrations.

These stress points motivate careful region sizing and partition strategies but do not alter the conceptual execution model.

5.12 Transition

With distributed execution defined for volumetric space, the next chapter addresses deterministic replay, auditability, and invariant commitments in the 3D setting, emphasizing continuity with the 2D formulation.

6 Deterministic Replay, Auditability, and Invariant Commitments (3D)

This chapter establishes deterministic replay and auditability for Qengine (3D) executed on a volumetric quansistor substrate. The mechanisms presented here are direct generalizations of the two-dimensional formulation and preserve identical semantic guarantees under increased spatial complexity.

6.1 Determinism in Volumetric Execution

Determinism remains a foundational property of Qengine (3D). Given:

- identical initial volumetric substrate configuration,
- identical mobile cell definitions,
- identical local update rules,
- identical message ordering at region boundaries,

the evolution of the three-dimensional system is fully determined.

The presence of volumetric neighborhoods does not introduce nondeterministic effects.

6.2 Replay by Re-Execution

Replay in Qengine (3D) is achieved through re-execution rather than trace replay. The system is restarted from the same initial configuration and evolved under the same update rules.

Correctness is evaluated by comparing invariant outcomes rather than procedural execution traces. This approach scales independently of execution duration and substrate volume.

6.3 Invariant Commitments in 3D

At selected time steps or epochs, each volumetric execution region produces an invariant commitment:

$$C_i^{(3)}(t) = \mathrm{Commit}(\mathcal{I}_i^{(3)}(t)),$$

where $\mathcal{I}_i^{(3)}(t)$ denotes the region's volumetric invariant summaries.

Commitments are compact, immutable, and independent of internal operator representations.

6.4 Commitment Granularity

Invariant commitments may be generated at different granularities:

- per volumetric region,
- per temporal window,
- per simulation phase.

Granularity selection balances audit resolution against storage and communication cost.

6.5 Global Audit Composition

Global audit verification is achieved by composing regional commitments:

$$C^{(3)}(t) = \bigotimes_i C_i^{(3)}(t).$$

The composition operator preserves ordering and identity while avoiding exposure of region-internal data. Global verification does not require centralized state aggregation.

6.6 Audit Without Volumetric State Disclosure

Invariant commitments do not disclose:

- operator parameters,
- mobile cell internal geometry,
- substrate configuration beyond invariant summaries.

This supports privacy preservation and selective disclosure even in large volumetric simulations.

6.7 Detection of Deviations

Any deviation from declared update rules, causal ordering, or message semantics manifests as a mismatch in invariant commitments.

In three dimensions, such deviations:

- are locally detectable,
- propagate through commitment composition,
- invalidate subsequent commitments.

Fault detection remains structural rather than procedural.

6.8 Temporal Integrity and Causal Chains

Invariant commitments form a temporal chain:

$$C^{(3)}(t_0) \to C^{(3)}(t_1) \to \cdots \to C^{(3)}(t_n),$$

establishing a causally ordered audit record.

The chain enforces temporal integrity without encoding execution history.

6.9 Distributed Audit Under Asynchrony

Regions may emit commitments asynchronously. As long as causal dependencies are respected, global audit composition remains valid.

This property is essential for execution on asynchronous, message-driven substrates with volumetric partitioning.

6.10 Stress Factors in 3D Auditability

Three-dimensional execution introduces quantitative stress factors:

- increased number of regions,
- larger boundary surfaces,
- higher invariant aggregation cost.

These factors affect performance but do not weaken audit guarantees.

6.11 Relation to QVM Semantics

Within the QVM framework, invariant commitments in Qengine (3D) function as:

- execution certificates,
- replay anchors,
- correctness witnesses.

The volumetric extension does not alter their semantic role.

6.12 Transition

With deterministic replay and auditability established for volumetric execution, the next chapter presents illustrative three-dimensional scenarios highlighting new classes of emergent phenomena enabled by the additional spatial dimension.

7 Illustrative Scenarios and Emergent Phenomena (3D)

This chapter presents illustrative scenarios highlighting emergent phenomena that arise specifically from three-dimensional quansistor substrates. The goal is to demonstrate how volumetric locality, topology, and interaction geometry enable qualitative behaviors unavailable in planar models.

The scenarios are conceptual and intended to clarify system behavior rather than to prescribe implementations or benchmarks.

7.1 Purpose of 3D Scenarios

Three-dimensional scenarios serve to:

- identify emergent structures unique to volumetric space,
- test the robustness of causal and audit semantics under increased connectivity,
- illustrate how operator-based dynamics scale with dimensionality.

All scenarios respect local update rules and no-signalling constraints.

7.2 Scenario A: Filament Formation

A set of mobile cells is initialized in a volumetric substrate with a weakly anisotropic invariant field. Cells follow local stability gradients while interacting indirectly through the substrate.

Observed behavior:

- formation of elongated filamentary structures,
- alignment along locally stable invariant ridges,
- slow migration and reconnection of filaments.

Such structures have no planar analogue and arise from volumetric connectivity.

7.3 Scenario B: Shell and Void Structures

Mobile cells operating in elastic mode are placed in a substrate containing localized invariant minima.

Observed behavior:

- aggregation of cells into hollow shells,
- formation of internal voids with distinct invariant signatures,
- persistence of shell structures under perturbation.

These configurations demonstrate volumetric coherence and internal topology.

7.4 Scenario C: Vortical and Circulatory Motion

Invariant fields with rotational components induce circulatory motion of mobile cells.

Observed behavior:

- stable or quasi-stable vortical trajectories,
- helical motion paths,
- coupling between rotation and translation.

Such behavior emerges without explicit angular momentum variables or forces.

7.5 Scenario D: Layered and Stratified Regions

The substrate is configured with layered invariant profiles varying along one axis.

Observed behavior:

- confinement of mobile cells to specific layers,
- barrier-like behavior between layers,
- slow cross-layer migration under sustained perturbation.

This scenario illustrates stratification as an emergent property of invariant structure.

7.6 Scenario E: Interaction Across Volumetric Boundaries

Cells are initialized near boundaries between volumetric execution regions.

Observed behavior:

- seamless migration across region interfaces,
- preservation of trajectories despite partitioning,
- absence of boundary-induced artifacts.

This validates distributed execution semantics in three dimensions.

7.7 Scenario F: Topological Defects and Singular Regions

Localized perturbations create regions where invariant structure is disrupted.

Observed behavior:

- formation of defect-like regions,
- trapping or repulsion of mobile cells,
- long-lived topological features.

Defects emerge as structural phenomena rather than encoded objects.

7.8 Scenario G: Multi-Cell Collective Dynamics

Dense populations of mobile cells interact indirectly through the volumetric substrate.

Observed behavior:

- collective motion patterns,
- spontaneous symmetry breaking,
- transient large-scale organization.

Collective behavior arises without global coordination or shared state.

7.9 Interpretation of Emergent Phenomena

The phenomena described here should not be interpreted as physical realism. Instead, they demonstrate how operator-based, invariant-driven dynamics support rich behavior under strict locality and causality constraints.

Three-dimensional space acts as an amplifier of topological and geometric structure rather than as a source of new semantics.

7.10 Relevance to QVM Exploration

These scenarios demonstrate that Qengine (3D):

- supports qualitatively new emergent behavior,
- preserves auditability under increased complexity,
- provides a stress-tested environment for QVM concepts.

They serve as reference patterns for future extensions and experimental studies.

7.11 Transition

The final chapter discusses the limitations of the 3D model, potential extensions, and open research problems introduced by volumetric quansistor field simulation.

8 Limits, Extensions, and Open Problems (3D)

This chapter delineates the conceptual and practical limits of Qengine (3D), outlines natural extensions of the volumetric model, and identifies open research problems introduced by three-dimensional quansistor field simulation. The intent is to clarify scope, prevent overinterpretation, and position Qengine (3D) as an experimental platform rather than a finalized system.

8.1 Conceptual Limits of the 3D Model

Qengine (3D) is an abstract computational construct. Despite its volumetric structure and emergent behavior, it does not claim to model physical spacetime, quantum fields, or biological organisms.

The three-dimensional substrate:

- is discrete and finite,
- operates on simplified operator abstractions,
- prioritizes causal clarity over physical realism.

Any resemblance to physical or biological systems should be interpreted as illustrative rather than explanatory.

8.2 Limits Introduced by Volumetric Complexity

Three-dimensional substrates introduce increased structural complexity:

- larger neighborhood sizes,
- higher invariant aggregation cost,
- increased inter-region boundary surfaces.

These factors constrain practical scalability and motivate careful design of sensing radii, region sizes, and invariant families.

8.3 CA-like Dynamics and Their Boundaries

Although Qengine (3D) exhibits CA-like characteristics, it does not inherit classical cellular automata properties such as state universality or symbolic rule completeness.

In particular:

- mobile cells are not state machines,
- update rules transform operators, not symbols,
- universality claims are intentionally avoided.

CA-like behavior should be understood as a structural analogy.

8.4 Determinism vs. Controlled Stochasticity

The default execution model of Qengine (3D) is fully deterministic. Introducing stochastic elements is possible but raises open questions regarding:

- reproducible randomness in distributed volumetric execution,
- stochastic invariant commitments,
- preservation of audit semantics under probabilistic updates.

Any stochastic extension must be explicit and externally parameterized.

8.5 Extension to Higher Dimensions

While Qengine (3D) already operates in volumetric space, the framework is not fundamentally limited to three dimensions. Higher-dimensional substrates may be considered as conceptual extensions.

Such extensions introduce:

- rapidly growing neighborhood complexity,
- reduced interpretability,
- diminishing returns for practical experimentation.

Higher-dimensional models are therefore of theoretical rather than practical interest.

8.6 Richer Invariant and Operator Families

The current formulation assumes relatively simple invariant families. Natural extensions include:

- multi-scale volumetric invariants,
- time-accumulated or history-sensitive summaries,
- adaptive invariant selection based on local dynamics.

These extensions raise questions about computational cost, sufficiency, and audit granularity.

8.7 Learning and Adaptive Behavior

An open research direction is the integration of learning or adaptation:

- modification of motion functionals,
- evolution of internal coupling graphs,
- invariant-driven feedback mechanisms.

Such mechanisms must remain compatible with locality, causality, and deterministic replay requirements.

8.8 Partitioning and Load Management

Volumetric partitioning introduces challenges in load balancing:

- uneven distribution of mobile cells,
- localized regions of high invariant complexity,
- migration-heavy boundary regions.

Addressing these challenges without introducing global coordination remains an open problem.

8.9 Relation to Other Distributed Dynamical Systems

Qengine (3D) intersects conceptually with several classes of systems, including:

- lattice field models,

- agent-based simulations,
- distributed dynamical systems.

However, its operator-first and invariant-centric semantics distinguish it from state-based formulations. Formal classification remains an open research topic.

8.10 Open Problems

Key open problems include:

- identification of invariant families that best capture meaningful volumetric dynamics,
- formal guarantees on invariant sufficiency for replay and audit,
- optimal strategies for volumetric partitioning under causal constraints,
- characterization of emergent structures unique to operator-based 3D substrates.

Progress on these problems will require both theoretical analysis and experimental simulation.

8.11 Conclusion

Qengine (3D) extends the quansistor field simulation framework into volumetric space while preserving strict locality, causality, and auditability. The model demonstrates that operator-based computation and invariant-driven dynamics scale to higher-dimensional substrates without introducing hidden coordination or state-based signaling.

As with its planar counterpart, Qengine (3D) is intentionally modest in scope. It serves as a testbed for exploring quansistor field computation, QVM semantics, and distributed execution under increased topological complexity.