Architectural Design and Engineering Protocols for High-Tier Military State-Machines in Autonomous Screeps Environments

The evolution of autonomous systems in the Screeps environment has moved beyond simple script-based reactions toward comprehensive military architectures that function as decentralized, yet highly coordinated, strategic entities. A top-tier military state-machine is defined not by the complexity of its individual unit code, but by the robustness of its hierarchical decision-making framework and its ability to mathematically quantify every tactical engagement. The paradigm shift required for modernization involves transitioning from legacy target-lock logic to an event-driven, hierarchical state-machine architecture that decouples strategic intent from tactical execution.

In the high-stakes ecosystem of the global control level (GCL) rankings, the margin between a successful siege and a total colony collapse often rests on a single tick of miscalculated healing throughput or a fatigue-induced break in squad cohesion. This report provides an exhaustive engineering breakdown of the protocols, mathematical models, and implementation steps necessary to develop and maintain a military system capable of competing at the highest tiers of autonomous warfare.

Foundational Command Hierarchy: The Overlord Framework

The most successful architectural pattern identified in the research is the hierarchical decoupling of colony management into specific layers: the Overmind, the Overseer, the Directive, and the Overlord. This structure ensures that high-level strategic goals do not interfere with the micro-optimizations required for individual unit performance.

The Overmind and Strategic Analysis

The Overmind serves as the global root object, providing persistent storage, global heap caching, and a unified interface for cross-room analytics. Within this layer, "Analysts" perform continuous polling of the world state, converting raw game data into actionable metrics. This sensing layer is critical because military state-machines cannot function effectively on raw sensory data alone; they require processed "Territory Intelligence".

Component	Responsibility	Performance Impact
Threat Analyst	Calculation of hostile DPT and heal capacity	High CPU, highly cacheable
Path Analyst	Multi-room routing and CostMatrix generation	Medium CPU, low cacheability
Economy Analyst	Monitoring tower energy and	Low CPU, medium cacheability

Component	Responsibility	Performance Impact
	lab throughput	
Strategic Selector	Priority-based task allocation across colonies	Medium CPU, low cacheability

The Overseer acts as the sensory brain of the colony, scanning for anomalous conditions such as an invasion, a structural breach, or a resource opportunity. Upon detection, the Overseer does not issue unit commands; instead, it instantiates Directives.

Directives as Goal Wrappers

Directives are functional wrappers for room flags that define a strategic goal for a specific location. By using flags as the basis for directives, the system achieves human-assisted automation: a developer can manually place a flag to trigger a siege, or the Overseer can automate this placement based on room scoring.
Directives are categorized by color codes, with "Red" typically reserved for military operations. A Red Directi[span_10](start_span)[span_10](end_span)ve on a hostile room acts as an attachment point for a Combat Overlord. This separation allows the system to remain modular; the logic for *deciding* to attack is entirely separate from the logic for *executing* the attack.

Overlords as Process Managers

The Overlord class replaces traditional "Role" systems by generalizing a set of related tasks into a single biological process. A Siege Overl[span_11](start_span)[span_11](end_span)ord handles the following lifecycle:

1. **Spawning:** Calculating the optimal body composition based on the target room's tower damage.
2. **Boosting:** Coordinating lab intents to apply T3 mineral compounds to the necessary parts.
3. **Logistics:** Ensuring the military unit reaches the target through safe highway routes.
4. **Tactics:** Driving the combat state-machine (Harass, Drain, Siege) once the creeps arrive.

Quantitative Defensive Logic: The Mathematical Perimeter

High-tier defense is a game of energy efficiency and burst-damage prevention. The military state-machine must transition between "Peacetime Maintenance," "Yellow Alert" (NPC or weak scouts), and "Red Alert" (Full-scale siege).

Tower Engagement Math and Energy Management

Towers are the primary tool for active defense, but they are economically expensive if used improperly. A tower's damage output decreases linearly as the distance to the target increases. The exact formula used by the Screeps engine is:
where Range is clamped between 5 and 20.

Range (r)	Condition	Attack (Hits)	Heal (Hits)	Repair (Hits)
r \le 5	Optimal Range	600	400	800

Range (r)	Condition	Attack (Hits)	Heal (Hits)	Repair (Hits)
5 < r < 20	Linear Falloff	600 - 30(r-5)	400 - 20(r-5)	800 - 40(r-5)
r \ge 20	Max Falloff	150	100	200

A high-tier defensive state-machine calculates the "Survivability Threshold" of a hostile creep before firing. If a hostile unit's heal-per-tick (HPT) exceeds the combined DPT of all towers at their current ranges, the state-machine should transition to a "Hold Fire" state to prevent energy draining. This is particularly critical because tower fire consumes 10 energy units regardless of effectiveness, making "Tower Draining" a viable strategy for attackers.

Passive Barrier Fortification and Layout

Structural defense relies on walls and ramparts. High-tier bots utilize the "Min-Cut" algorithm to determine the minimum number of ramparts needed to seal a base footprint. A common engineering error is placing ramparts too close to internal structures. Best practices dictate a 3-tile buffer to prevent rangedMassAttack from damaging internal buildings through the rampart.

RCL	Wall Limit (Hits)	Max Towers	Strategic Shift
3	N/A	1	Transition from spawn-defense to initial walling
5	Controller Level dependent	2	Introduction of Terminal energy sharing
7	Controller Level dependent	3	Introduction of Lab-boosted sorties
8	300,000,000	6	Absolute energy throughput and Nuke defense

During a Red Alert state, the machine prioritizes "Repair Spam" over "Attack" if tower fire cannot secure a kill. A room can sustain approximately 60,000 hits of repair per tick if the economy is sufficiently robust, making well-defended RCL 8 rooms nearly impossible to crack with unboosted forces.

Automated Safe Mode Trigger Protocols

Safe Mode is a non-renewable resource (one activation per level) and must be guarded by sophisticated trigger criteria. The "Fail-Safe" state should only be entered under the following conditions:

1. **Critical Breach:** A rampart protecting a core structure (Storage, Spawn, Terminal) has fallen below 10,000 hits and a hostile creep with WORK or ATTACK parts is adjacent.
2. **Pathfinding Threat:** The ThreatAnalyst finds a valid path from a hostile creep to the Spawn that does not pass through a rampart.
3. **Worker Attrition:** Hostiles have destroyed more than 50% of the colony's worker creeps, and energy sharing via Terminals is offline.
4. **Adjacency Check:** A hostile creep with WORK parts is adjacent to the room's Controller.

Offensive State-Machines: The Mechanics of Seizure

Offense in Screeps is significantly more resource-intensive than defense, requiring an estimated

3:1 economic advantage for success. A high-tier offensive state-machine operates through a series of tactical transitions: Scouting, Harassment, Draining, and Siege.

Scouting and Strategic Selection

Before an attack is launched, the Overseer places a Purple Scout Directive. The scout creep's goal is to record the following metadata in the global memory:

- **Wall Thickness:** The average and minimum hits of the enemy ramparts.
- **Tower Energy:** Whether the defender has an active tower-refilling script.
- **Boost Availability:** Presence of Labs and whether they are stocked with XGHO2 (Tough) or XLHO2 (Heal).

This information determines the state transition: if towers are unrefilled, transition to Tower Draining; if walls are thin, transition to Siege; if the opponent is high-tier, transition to Room Quarantining.

The "Drainer" State-Machine

Tower draining is the most cost-effective offensive strategy. The state-machine for a drainer unit follows a specific oscillating logic:

1. **Approach:** Move to range 1 of the room exit.
2. **Step In:** Move one tile into the hostile room, ideally at range \ge 20 from all towers.
3. **Soak:** Wait for the end-of-tick tower processing to apply damage.
4. **Step Out:** On the next tick, move back into the safe room.
5. **Recover:** Receive healing from a stationary healer in the safe room until hits are at 100%.

The goal is to force the defender to spend 10 energy per shot while the attacker's healing costs only the time (ticks) of the unit. Against bots with poor terminal logic, this will eventually trigger an economic collapse.

The Siege and Destruction FSM

A siege is initiated once the defender's energy is low or the attacker has achieved Tier 3 boosting capability. The machine spawns "Breakers" (boosted work creeps) and "Medics".

Role	Core Parts	Primary Boost	Tactical Goal
Breaker	TOUGH, WORK, MOVE	XZHO2, XGHO2, XH2O	Dismantling ramparts and spawns
Medic	TOUGH, HEAL, MOVE	XZHO2, XGHO2, XLHO2	Synchronized healing of the Breaker
Destroyer	TOUGH, ATTACK, MOVE	XZHO2, XGHO2, XUH2O	Elimination of hostile sortie creeps

The state-machine transitions from TRAVEL to SIEGE_TARGET once the breaker is adjacent to a wall. It is a best practice to use dismantle() over attack() for structures, as WORK parts provide 50 damage per part compared to the 30 damage of ATTACK parts.

Unit Coordination: The Quad Movement Engine

In the high-tier meta, single creeps are easily countered by random focus fire or specialized melee defenders. To overcome this, top-tier bots implement "Quads"—synchronized groups of

four creeps moving as a 2x2 unit.

Synchronized Movement and Traffic Resolution

A quad must move in lockstep. If any member lags due to fatigue, the formation breaks and the group's collective heal power is halved. Implementation of a quad movement engine requires a custom pathing solution:

1. **Leader Assignment:** One creep is designated as the "Master," usually the top-left unit of the quad.
2. **Custom CostMatrix:** The pathfinder considers any tile that would place *any part* of the 2x2 footprint in a wall to be impassable. This typically means marking every tile adjacent to a wall (top and left) as cost 255.
3. **Fatigue Synchronization:** The squad's move intent is only issued if *all* members have zero fatigue. If one member enters a swamp, the entire squad waits for that unit to recover.

Wavefunction Collapse for Shoving

Traffic jams are the primary cause of quad failure inside a base. To resolve this, high-tier bots use a variant of the "Wavefunction Collapse" algorithm for traffic management.

- **Intent Collection:** Every creep in the room submits an array of valid move targets (with their current position as the lowest priority).
- **Priority Tiers:** Military quads are given top-tier priority, followed by defenders, haulers, and then static harvesters.
- **Resolution:** The engine iterates through the highest-priority intents. If a quad's target tile is occupied by a lower-priority creep, the system "shoves" the worker into an adjacent valid tile, effectively clearing a path for the military formation.

Mathematics of Chemical Warfare: The Tier 3 Advantage

The delta between an unboosted creep and a Tier 3 boosted creep is an order of magnitude, not a marginal increase. A military state-machine must integrate lab logic as a prerequisite for any offensive state transition.

Stacking Boosts and Effective Hit Points

Tier 3 TOUGH boosts (XGHO2) reduce incoming damage by 70%. This effectively multiplies the hit points of that body part by 3.33.
For a Tier 3 boosted part:
This stacking effect is the only way a creep can survive the focus fire of six RCL 8 towers (6 \times 600 = 3,600 \text{ DPT}). Without boosts, even a 50-part creep would be destroyed in 2 ticks. With boosts, the effective HP of 13 TOUGH parts is 4,330, which allows a single medic to fully recover the damage of those towers in the same tick if they have 27 Tier-3 HEAL parts (27 \times 48 = 1,296 \text{ HPT} adjusted for tough parts).

Part Type	Unboosted Effect	T3 Boosted Effect	Multiplier
ATTACK	30 dmg	120 dmg	4.0x
RANGED_ATTACK	10 dmg	40 dmg	4.0x
HEAL	12 hits	48 hits	4.0x
WORK (Dismantle)	50 dmg	200 dmg	4.0x
MOVE	-2 fatigue/tick	-8 fatigue/tick	4.0x

The "Pre-healing" and Synchronization Logic

In the Screeps engine, intents are processed in batches. A major engineering challenge is the timing of heal() versus incoming attack(). If damage is applied first and reduces a creep to 0 hits, any subsequent heal intent on that creep will fail, as the unit is already dead.
Top-tier state-machines implement "Pre-healing": healers call heal() on their squad members every tick *regardless* of their current health. This ensures that if the creep takes damage during the tick resolution, it immediately begins recovery. In a quad, each member should prioritize healing the squad member with the lowest *projected* health based on tower focus fire predictions.

System Implementation: Coding a Modular Military FSM

The implementation of a top-tier state-machine requires a move away from "Role" modules toward "State Classes" that implement a standard interface. This allows the bot to decouple the *behavior* of the unit from the *entity* itself.

The State Class Pattern

Instead of a giant switch statement in a role file, each state is represented by an object with a run() and transition() method.

```
class EngageState {
    run(creep, context) {
        // Find best position using kiting logic
        if (creep.pos.isNearTo(context.target)) {
            creep.attack(context.target);
        } else {
            creep.moveTo(context.target);
        }
    }

    transition(creep, context) {
        if (creep.hits < creep.hitsMax * 0.5) return 'RETREAT';
        if (!context.target) return 'IDLE';
        return 'ENGAGE';
    }
}
```

This pattern prevents "obscure logic bugs" by making transitions explicit. A unit's state is stored

in its memory, and the main loop simply instantiates the corresponding state class and calls run().

Intent Order and Simultaneous Actions

A military unit should maximize its per-tick value by combining non-exclusive actions. For example, a single creep can perform move, attack, and rangedHeal in the same tick if it possesses the necessary parts.

Action A	Action B	Compatible?	Constraint
attack()	heal()	No	Same action pipeline
rangedAttack()	heal()	Yes	Different pipelines
move()	attack()	Yes	Standard kiting requirement
dismantle()	attack()	No	Same action pipeline
build()	repair()	No	Same pipeline

The state-machine's "Action Resolver" should verify these combinations every tick. High-tier bots use a "greedy" resolver that attempts to execute every available military intent, starting with the most critical (Healing) and ending with secondary effects (Ranged harassment).

Advanced Tactical Maneuvers: Nukes and Power Creeps

In the late game (RCL 8 and beyond), traditional creep combat reaches a stalemate due to repair throughput. Breaking this stalemate requires the integration of non-standard military assets: Nukers and Power Creeps.

Nuke Orchestration logic

Nukes take 50,000 ticks to land and are highly visible. They are rarely used to destroy units; instead, they are used for "Infrastructure Denial."

- **Safe Mode Resetting:** If a defender activates Safe Mode (20,000 ticks), a nuke launched *after* the activation will land after the Safe Mode ends, providing a 200-tick window where the defender cannot Safe Mode again.
- **Structural Shock:** Targeting the Terminal and Storage simultaneously forces the defender to spend vast amounts of energy on emergency repairs, often stalling their upgrader pipeline.

Nuke Metric	Value	Strategic Implication
Flight Time	50,000 ticks	Requires long-term forecasting
Center Damage	10,000,000 hits	Destroys everything but high-tier ramparts
Outer Damage	5,000,000 hits	Forces massive repair costs
Safe Mode Lock	200 ticks	Primary window for siege entry

Power Creeps as Force Multipliers

Power Creeps introduce abilities that disrupt standard state-machine calculations.

- **Exhaust:** Increases fatigue on hostiles, effectively "pinning" a quad in range of towers.

- **Jam:** Forces a creep to repeat its last action, disrupting kiting or retreating state transitions.
- **GeneratePortal:** Linking two Power Creeps allows for the near-instant projection of military force across sectors, bypassing the standard multi-room travel state.

A top-tier military machine must account for these "Debuff" states in its unit FSM. A unit that is Exhausted must immediately transition to an Emergency Shield state, requesting all available nearby healers to prioritize it until the effect wears off.

Implementation Steps for Top-Tier Military Modernization

The roadmap for implementing a modernized military state-machine within the grgisme/screeps codebase involves a systematic overhaul of the sensing and action layers.

Phase 1: The Sensing Layer (Tick 0 - 1000)

Establish the "Analyst" infrastructure. This involves creating singleton modules that monitor room vision and update global metadata.

1. **Hostile Catalog:** Maintain a list of all active players in range and their observed RCL/boost tiers.
2. **Combat Simulation:** Develop a module that can take two creep bodies and simulate their interaction over 10 ticks. This "Battle Simulator" is the backbone of offensive decision-making.
3. **Min-Cut Perimeter:** Automate base layout checks to ensure 100% rampart coverage with optimal buffers.

Phase 2: The Tactical Machine (Tick 1000 - 5000)

Move away from per-creep role logic and toward group-based state-machines.

1. **Quad Movement Engine:** Implement the 2-wide CostMatrix and cohesion-wait logic.
2. **Traffic Reconciler:** Integrate a wavefunction collapse resolution for move intents to prevent squad breakage.
3. **Pre-heal Logic:** Ensure all military healers call heal() on every tick, using projected damage as the targeting priority.

Phase 3: The Strategic Layer (Tick 5000+)

Implement the Overlord/Directive hierarchy for fully autonomous warfare.

1. **Expansion Quarantining:** Upon detecting a hostile expansion within range 2 of the colony, automatically place a Harassment Directive.
2. **Economic Denial:** Use the Tower Draining FSM to target high-energy players, forcing them into a deficit state.
3. **Nuke Interception:** Develop the "Emergency Repair" FSM that detects incoming nukes and prioritizes rampart height on those specific tiles.

Causal Implications of High-Tier Military Design

The engineering data suggests several deep-order insights into the nature of top-tier play. First, military success is fundamentally an economic problem. A bot with superior kiting code will still lose to a bot with superior lab-automation and energy-sharing logic, as the latter can afford to sustain infinite unboosted attrition or a finite T3 boosted steamroll.
Second, the game's deterministic nature means that the "best" move is usually discoverable through simulation. High-tier machines spend significant CPU on "What-If" scenarios: if the quad moves to tile (X,Y), what is the maximum damage it can take, and is the current HPT sufficient to survive?. If the simulation shows a survival chance of <100\%, the state-machine *must* transition to RETREAT. This "Knife-Edge" survival is what makes Screeps military logic an engineering discipline rather than a typical game AI challenge.
Finally, the trend in Screeps warfare is moving toward "Centralized Decision, Decentralized Execution." The strategic directives are issued from the colony center, but the tactical quads must have enough local autonomy to resolve traffic and intent conflicts in real-time.

Conclusions and Engineering Recommendations

The development of a high-tier military state-machine is an iterative process of quantification and abstraction. To satisfy the requirements for a modernized, top-tier system within the context of the user's focus, the following final recommendations are made:

1. **Quantify Defensive Thresholds:** Implement a room-wide DPT vs HPT check. Never fire towers if the enemy's boosted heal capacity exceeds the possible damage. This is the single most important fix to prevent tower draining.
2. **Standardize the Overlord Pattern:** All military operations should be encapsulated in Overlord objects that manage the entire lifecycle from lab-boosting to extraction. This eliminates "sloppy bugs" where units forget to boost or arrive with mismatched TTL.
3. **Master the Quad:** Single-unit combat is obsolete at RCL 8. The quad move engine is a non-negotiable prerequisite for high-tier sieging.
4. **Automate the Safe Mode fail-safe:** Implement path-based threat detection to trigger Safe Mode. Relying on a fixed "hit threshold" is too slow against T3 boosted dismantlers that can destroy a 10M rampart in fewer than 100 ticks.
5. **Implement Pre-healing Pass:** Modify the intent resolution system to call heal() at the start of the tactical loop, prioritizing projected health deltas over current health deltas. This negates the "Alpha-Strike" advantage of defenders.

By strictly adhering to these architectural and mathematical protocols, a Screeps bot transitions from a collection of defensive scripts into a dominant military force capable of project power across the global map. The future of autonomous warfare belongs to those who treat their codebase as a weapons-grade engineering system.

Works cited

1. Screeps #1: Overlord overload - Ben Bartlett, https://bencbartlett.com/blog/screeps-1-overlord-overload/ 2. Encouraging more combat at high GCL | Screeps Forum, https://screeps.com/forum/topic/2809/encouraging-more-combat-at-high-gcl 3. Workflow tips

and prioritization for new players? | Screeps Forum, https://screeps.com/forum/topic/2556/workflow-tips-and-prioritization-for-new-players 4. QP/C: State Machines - Quantum Leaps, https://www.state-machine.com/qpc/srs-qp_sm.html 5. Damage order of operations with partialy damaged boosted tough and pre-healing | Screeps Forum, https://screeps.com/forum/topic/2801/damage-order-of-operations-with-partialy-damaged-boosted-tough-and-pre-healing 6. Screeps #12: Strategic Directives | Field Journal, https://jonwinsley.com/notes/screeps-strategic-directives 7. Screeps #14: Decision Making - Field Journal, https://jonwinsley.com/notes/screeps-decision-making 8. Screeps #3: State of the Automated Union | Ben Bartlett, https://bencbartlett.com/blog/screeps-3-state-of-the-automated-union/ 9. Automating Base Planning in Screeps – A Step-by-Step Guide, https://sy-harabi.github.io/Automating-base-planning-in-screeps/ 10. MATH.md - ryanrolds/screeps-bot-choreographer - GitHub, https://github.com/ryanrolds/screeps-bot-choreographer/blob/main/MATH.md 11. Quad (high volume creep attack) · Issue #69 · JonathanSafer/screeps - GitHub, https://github.com/jordansafer/screeps/issues/69 12. Boosts - Screeps Wiki, https://wiki.screepspl.us/Boosts/ 13. Offensive Strategy : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/7d6w2t/offensive_strategy/ 14. Great Filters - Screeps Wiki, https://wiki.screepspl.us/Great_Filters/ 15. Defending your room | Screeps Documentation, https://docs.screeps.com/defense.html 16. Why are the towers broken? :: Screeps: World General Discussions - Steam Community, https://steamcommunity.com/app/464350/discussions/0/1699415798767961303/ 17. Screeps #21: Patrolling the Perimeter - Field Journal, https://jonwinsley.com/notes/screeps-patrolling-perimeter 18. Best Attack Strategy? | Screeps Forum, https://screeps.com/forum/topic/2995/best-attack-strategy 19. How do you deal with healer attacks? : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/4z8bz3/how_do_you_deal_with_healer_attacks/ 20. StructureTower - Screeps Wiki, https://wiki.screepspl.us/StructureTower/ 21. Tower Damage | Screeps Forum, https://screeps.com/forum/topic/1097/tower-damage 22. Automated Base Planning | Screeps Tutorial - YouTube, https://www.youtube.com/watch?v=YcruUDbqa7E 23. Combat - Screeps Wiki, https://wiki.screepspl.us/Combat/ 24. Controller | Screeps Wiki - Fandom, https://screeps.fandom.com/wiki/Controller 25. Auto Safemode - Code for new players :: Screeps: World Help - Steam Community, https://steamcommunity.com/app/464350/discussions/5/152390648081885882/ 26. does anyone automate safe mode? if so what triggers it to activate? : r/screeps - Reddit, https://www.reddit.com/r/screeps/comments/662flg/does_anyone_automate_safe_mode_if_so_what/ 27. Boosted creep bodyparts - forum - Screeps, https://screeps.com/forum/topic/3034/boosted-creep-bodyparts 28. RCL 6 Defense From Duo | Screeps Combat Analysis - YouTube, https://www.youtube.com/watch?v=g-lvDUyaNmM 29. JonathanSafer/screeps: Screeps AI - GitHub, https://github.com/JonathanSafer/screeps 30. Creep | Screeps Wiki | Fandom, https://screeps.fandom.com/wiki/Creep 31. Screeps #25: Arena - Grouping Up - Field Journal, https://jonwinsley.com/notes/screeps-arena-grouping-up 32. Traffic Management | screeps-cartographer, https://glitchassassin.github.io/screeps-cartographer/pages/trafficManagement.html 33. screeps-cartographer, https://glitchassassin.github.io/screeps-cartographer/ 34. Cartographer is an advanced (and open source) movement library for Screeps - GitHub, https://github.com/glitchassassin/screeps-cartographer 35. screeps-cartographer/README.md

at main - GitHub,
https://github.com/glitchassassin/screeps-cartographer/blob/main/README.md 36. Overhealing vs. strictly healing past damage: fixing inconsistencies based off of intent order | Screeps Forum,
https://screeps.com/forum/topic/319/overhealing-vs-strictly-healing-past-damage-fixing-inconsistencies-based-off-of-intent-order 37. Best Practices for State Machine Implementation : r/Unity3D - Reddit,
https://www.reddit.com/r/Unity3D/comments/1b6973c/best_practices_for_state_machine_implementation/ 38. How to transition between states and mix states in a finite state machine?,
https://gamedev.stackexchange.com/questions/7906/how-to-transition-between-states-and-mix-states-in-a-finite-state-machine 39. A functional boilerplate for screeps.com with creeps based on the fsm pattern - GitHub Gist, https://gist.github.com/fd6aae59f4193c63c8c3f28b0aeb51e2 40. Screeps #20: The Great Purge - Field Journal,
https://jonwinsley.com/notes/screeps-great-purge 41. Simultaneous execution of creep actions - Screeps Documentation, https://docs.screeps.com/simultaneous-actions.html 42. Are any Actions of Screeps mutually exclusive to eachother? - Arqade - Stack Exchange,
https://gaming.stackexchange.com/questions/200912/are-any-actions-of-screeps-mutually-exclusive-to-eachother 43. PvP Game discussion | Screeps Forum,
https://screeps.com/forum/topic/473/pvp-game-discussion/35 44. Super Structures and Creeps Idea. | Screeps Forum, https://screeps.com/forum/post/11239 45. [Power] Power creep ideas | Screeps Forum, https://screeps.com/forum/topic/388/power-power-creep-ideas 46. Calculating the probability of winning in a fight given the characters' health, damage and healing? : r/math - Reddit,
https://www.reddit.com/r/math/comments/8typwz/calculating_the_probability_of_winning_in_a_fight/