Cogmind Manual version: Beta 13

This file contains information regarding gameplay rules. For information about running the game see COGMIND-README.txt.

Intro

This manual is not an exhaustive description of everything that may happen in the game. Rather it provides an overview of the core rules necessary for a fundamental understanding of how to play, while also explaining some features that may not be made explicitly known during normal play.

As for features that aren't described herein, discovery is all part of the fun!

Other sources of information include the help screens (for commands), tutorial messages that trigger the first time an essential mechanic is encountered, and context help available through both the status and item data pages (left-click on a data entry, or use the Up/Down arrows and press Enter).

(Note that this manual is accessible via both the in-game UI, F1->3, and as a text file in the game directory: manual.txt. The latest version can also always be found online at gridsagegames.com/cogmind/manual.txt)

Using this Manual

It is not recommended for beginners to read through this entire manual, which could seem overwhelming at first. What you really need to know is covered by the automatic tutorial, context help, and basic item descriptions (e.g. matter). You can become a good intermediate player without even reading the manual, but coming back later to fill in any knowledge gaps could be an important step towards becoming an expert.

However, the next section ("A Roguelike") is recommended reading for everyone, whether familiar with roguelikes or new to the genre. And if you are having trouble with survival, or hope to improve your skills much more quickly, the following section ("Survival Tips") is highly recommended as well.

Note this manual is spoiler free.

A Roguelike

Cogmind belongs to an old genre of games called roguelikes, many of which traditionally share certain unique qualities. There's no need to describe them in full, but for any of you who are unfamiliar with the genre there are a couple that are important to be aware of.

Permadeath

Death is permanent. There is only one "save slot", used to save your progress when you exit, so returning later restarts right where you left off. Dying means you'll be starting over from the beginning, hopefully having learned enough from your previous experience to increase your chances of survival.

Turns

Action in Cogmind is played out in turn-based fashion, though individual turns may be carried out extremely quickly. That said, you can spend as long as you like thinking about what to do next, because the game waits for you to take your turn before allowing any other robots to act. All the robots will take their own turns in order, as the game proceeds at the pace you determine by how fast you enter commands. This system is explained in more detail in the manual section on Time.

Important Differences

But if you're coming from other traditional roguelikes, know that there are also quite a few aspects of Cogmind that might throw you off!

For one the message log has been deemphasized. It's not necessary to follow the log to figure out what's going on in combat; instead the map provides you with all the important information. When attacking a target, for example, the only truly significant hits are those on its core, and if you hit the core a little number pops up showing their remaining core integrity percent. Most robots are destroyed by just a couple core hits.

Cogmind also focuses on ranged combat, and the projectile mechanics are unique among roguelikes. Know that attacks will tend to miss a lot unless you design a build focused on accuracy or do more fighting at close range. Accuracy takes into account many factors, which you can read about in the manual under Combat (Hit Chance) to optimize your tactical superiority. If you see your projectile stop at the target's location then it hit somewhere on that target, though maybe not the core. This system is explained in more detail in the manual section on Combat, under Coverage.

Your own core is extremely resilient compared to other robots, so it is literally impossible for you to be "one-shotted," or even killed quickly, especially if you keep yourself covered with parts at all times. Use this knowledge to your advantage! It's more likely that you'll succumb to attrition over a longer period, or lose due to a series of mistakes made earlier than where you finally go down. But this resilience also gives you ample opportunity to rebuild and make epic comebacks, which are quite common in Cogmind.

Unlike in other roguelikes, rarely will you ever want to fully explore a map. Many are quite expansive, there is no XP to gain from taking on everything you see, and if your build is looking good you'll generally want to seek an exit and move on to avoid unnecessary damage.

Most importantly, remember that it's rarely just you vs. what you see in the local area! To an extent the world is alive on a macro level, and paying close attention to your surroundings is vital for survival. Observant players will learn to both intuit elements of the map layout and foresee dangerous situations.

There are many other significant differences between the average roguelike and Cogmind, but the rest are best discovered on your own.

Survival Tips

If you're having trouble surviving the early floors, below is a collection of spoiler-free beginners tips that will greatly improve your effectiveness.

Note some of these tips can and will be ignored by advanced players for various reasons, and you'll eventually be discovering your own play style and strategies, but in general terms they apply for new players and will help get you past early-game areas to the real challenges and fun stuff!

Items

Prefer higher-rating parts. At first the number of parts will seem overwhelming, especially if you feel the need to compare every detail to decide which you want to keep or use. While the item stat comparison info will help here (seen as red/green numbers after opening info for two items in a row), if you're having trouble deciding between two items or don't want to examine all the stats, there's a much faster method: Just compare their ratings. Ratings appear near the top of item info, and also as a number displayed next to the item name in the on-map labels, making it fairly easy to get a general idea of whether an item is worth it without even opening the info page. Prototypes, indicated with a *, are better than regular items of the same rating.

Builds

Legs are the best form of starter propulsion. They move fast enough, have a relatively high integrity, and you can salvage spares off enemy Grunts fairly easily. Avoid using wheels if possible. And generally avoid mixing propulsion, since you can't activate multiple types at once, anyway.

The safest first evolution is two propulsion slots, since an extra two legs will help block shots, and you won't have any problems carrying any amount of mass you want. If you're running out of propulsion parts to fill your slots, as long as you're not hovering or flying you should temporarily keep even deactivated propulsion attached (especially legs and treads), because they don't count against movement but can block shots. For later evolutions pick whatever you want, except Power is not usually necessary until you find yourself frequently running low on energy.

Almost all robots have static loadouts, therefore if you're looking for certain parts to put together or maintain the kind of build you want, one approach is to hunt down robots you know to use them. For example, Watchers carry sensors, and Sentries use treads and armor.

Inventory Management

Keep a full inventory whenever possible! Even if you don't need surplus parts right away, before long you'll need replacements as yours are destroyed. There are usually parts lying around, especially after a battle, and since inventory contents do not count against movement or have any other negative side effects, leaving it partially empty is an unnecessary waste.

Also don't leave empty part slots when possible, because filling them, even with parts you don't need or won't use, will at least help block incoming fire and protect your core.

One of the safer ways to play is to carry at least one Storage Unit, preferably a larger one, to allow for carrying more spare parts in inventory. Make sure you have an extra power source and weapons, and probably spare propulsion, too. The 't' key, or button under your inventory, is useful for type-sorting your inventory to help ensure that you have the right amount and types of spares.

Combat

Avoid unnecessary combat. It's fun to shoot everything, but until you have a better grasp of the consequences, and how to excel at combat, fighting too much only puts you at a disadvantage. That said, once you've engaged enemies in combat, try not to run as you'll probably end up running into more enemies and the situation will continue escalating.

Positioning is extremely important. At the start of combat against multiple enemies, reposition to a less-exposed position, preferably diagonal to the inside of a doorway, or a narrow corridor that can serve as a bottleneck. It's even worth taking damage in the process of repositioning, as you would likely take even more damage continuing a fight from a bad location. Fighting in the open is dangerous when not properly equipped to do so, but if you have a launcher, then by all means do it in the open!

Retreating to fight in more isolated areas also has the advantage of avoiding additional patrols that might happen upon a battle in progress. Even when not in combat, hugging walls while moving is a good way to stay less exposed and make access to nearby doors and corridors easier when necessary.

Attach replacement parts even in the middle of battle, in order to better protect your core from damage, and stay at maximum effectiveness. Attaching and swapping parts is a quick action.

Other Robots

Shoot at Scavengers once and they'll go away. Sometimes this is a good idea even in the middle of combat, if you want to have any loot left to salvage when it's over.

If a Watcher doesn't send out an alert, that means there are no enemies nearby, otherwise be prepared!

Destroying Haulers might provide you with relatively powerful parts to attach. Other non-hostile bots are also good targets to salvage if you're low on parts or need spares to fill your inventory. They won't be good, but you never want to be without a backup power source, for example.

Hacking

Hacking machines is not too important for beginners, but as you start to get into it, be aware that if you are fully traced it's likely more units will be dispatched to that area to investigate. If you don't want company, always quit hacking a machine before the trace ends! Being initially "detected" is fine, so you can always safely hack a new machine until at least that point.

Difficulty

Lowering the difficulty setting is not likely to help teach fundamental strategies, but an alternative approach would be to set a manual seed (in the options menu) to some phrase, and repeatedly play the same world to focus on getting the hang of the mechanics, rather than dealing with so many unknowns. When you feel more confident, turn off the manual seed.

(This approach is not recommended right away. Instead, consider it after having read the other tips and tried to apply them in regular random worlds.)

More Info

For other tips, both spoiler-free and otherwise, see the Strategy section of the forums. Links to more resources will continue to be added there in the future.

In the end, know that once you're familiar with all the mechanics and possible strategies, Cogmind is not a heavily luck-based game, and skilled players streak wins repeatedly.

Difficulty

The first time Cogmind starts up, you're presented with a pre-game menu for difficulty selection. Your chosen setting can later be modified in the options menu, although the effect won't kick in until a new run is started.

Cogmind is technically designed for Rogue mode, carefully balanced to provide a fun yet challenging roguelike experience which can be reliably overcome given sufficient experience and skill, but it will put

your strategic and tactical analysis skills to the test. It is extremely hard, especially for new players.

Naturally some players simply don't have the time or inclination to strive for mastery, thus alternative modes are available that tweak multiple aspects of the game to make survival easier.

That said, keep in mind that Cogmind relies on a fairly tight design to begin with, and giving you the upper hand via easier settings somewhat destabilizes that design, resulting in a less consistent difficulty curve. Some areas might be significantly easier, almost trivialized, while others remain relatively difficult due to the nature of the world and its mechanics. Also note that even within Rogue mode the world contains places to go and things to do which make the game easier, or even more difficult, if that's your thing!

So that it's more apparent from screenshots which difficulty setting a given player is using, the parts list slot type divider lines in the HUD appear in a different color for each. The stat screen shown at the end of a run also uses different highlight colors.

Important: The same world seed will produce different results in each difficulty setting!

Rogue

Rogue mode is the original intended way to play Cogmind, but it's definitely not for everyone. Although every run is winnable, it can take many dozens of hours to reach that level of skill due to the numerous local and global factors working against you, as well as time invested in exploring the wide variety of mechanics and systems you can take advantage of.

This mode reflects the difficulty of a typical traditional roguelike, hard to master but very satisfying as you achieve new milestones, and especially once you win.

* Permadeath is enforced (no optional quicksave/load feature)
* There are no adjustments for this mode, as it is the standard against which other modes are gauged

Adventurer

Adventurer mode is still fairly challenging, with a range of minor buffs and mechanical adjustments taking the edge off the unforgiving Rogue mode. Getting far will still take experience and persistence, but it's not quite as intense. Adventurers are Rogues in training. Adjustments:

* Quicksave/load support
* 20% base resistance to all damage types
* +20% base accuracy (both ranged and melee)
* Enemies alerting nearby allies reveal those allies' positions
* Nearby garrisons responding to call for help are revealed on map
* Effect of allies on alert level halved
* Disabling machines has less of an impact on alert level
* Traveling to a new area/map lowers alert level more than usual
* Alert level decay from passing turns doesn't have diminishing returns
* Each map containing Heavies converts 1 of them to a Sentry
* Heavy class Active Sensor Suites are 25% less sensitive
* Cargo convoys have 1 less ARC escort
* Part corruption effect reduced by 25%
* +10% to all direct and indirect hacks at interactive machines
* Minimum of 2 indirect database hacks before potential floorwide lockdown
* No Operators spawn at depth -10
* No Operators spawn within view of starting location below depth -8
* +25% to all Relay Coupler value
* Z-hack mapwide cutoff threshold higher

* [RPGLIKE mode] Start with 1,000 extra XP
* [RPGLIKE mode] Gain XP more quickly if fell behind the level curve
* [RPGLIKE mode] Protomatter has somewhat higher value on later maps
* [Pay2Buy mode] Start with 5,000 extra CogCoins
* [Polymind mode] Minimum amount of Protomatter dropped increased by 20%
* [Polymind mode] Suspicious 0b10 robots raise suspicion 25% more slowly

Explorer

Explorer mode is primarily for those interested in discovering more of the story and exploring various areas of the world that are otherwise much more difficult to reach in other modes. Or just running around shooting things.

Explorer mode is quite easy for those familiar with Cogmind mechanics, certainly too easy by roguelike standards. Yes you can still lose, and certain parts of the world are still going to be pretty dangerous, though with just a little experience it will be easier to avoid them (if you want to) because you'll automatically know where all exits lead. Adjustments:

* Quicksave/load support
* 35% base resistance to all damage types
* +30% base accuracy (both ranged and melee)
* Enemies alerting nearby allies reveal those allies' positions
* Nearby garrisons responding to call for help are revealed on map
* Effect of allies on alert level cut to one-third
* Disabling machines has a much lower impact on alert level
* Traveling to a new area/map lowers alert level much more than usual
* Alert level decay from passing turns doesn't have diminishing returns
* Alert level decays 50% faster as turns pass
* Each map containing Heavies converts 2 of them to Sentries
* Heavy class Active Sensor Suites are 50% less sensitive
* Heavy class sensor-based reinforcements limited to one active squad at a time
* Lower chance of random hostile encounters (most notably in mines/caves)
* Hostile branch encounters (e.g. in caves/mines) cannot appear directly in spawn area
* -1 to all patrol squad sizes (stacks with garrison effect)
* -1 to number of garrisons per floor (cannot reduce to 0, however)
* Fabricators do not automatically trace and investigate BUILD commands hacked without an Authchip
* Cargo convoys have 1 less ARC escort
* Always know active cargo convoy route
* Exit destinations automatically revealed on sight, or when found via Terrain Scanners
* Robot salvage in caves does not self-destruct
* Part corruption effect reduced by 50%
* Evolve 1 extra slot on exiting Scrapyard
* Scrapyard contains random useful utilities
* Scrapyard Storage Unit is Med. rather than Sml.
* Items nearby your spawn in main complex maps replaced by random items more likely to be useful given your current state
* No dormant Specialists stationed inside DSFs
* +20% to all direct and indirect hacks at interactive machines
* Unlimited free guaranteed indirect database hacks before potential floorwide lockout
* No central database lockdowns for indirect record hacking
* All direct Record hacks at terminals have an automatic 100% chance to succeed
* No Operators spawn at depth -10
* No Operators spawn within view of starting location below depth -7
* No Alarm Traps below depth -7
* +50% to all Relay Coupler value

* Z-hack mapwide cutoff threshold significantly higher
* [RPGLIKE mode] Start with 3,000 extra XP
* [RPGLIKE mode] Gain XP more quickly if fell behind the level curve
* [RPGLIKE mode] Protomatter has higher value on later maps
* [Pay2Buy mode] Start with 10,000 extra CogCoins
* [Polymind mode] Minimum amount of Protomatter dropped increased by 50%
* [Polymind mode] Frequency of Protomatter drop preference increased by 25%
* [Polymind mode] Suspicious 0b10 robots raise suspicion at half normal rate

Saving/Loading

Cogmind normally saves your progress when you exit the game with an unfinished run, then automatically loads back to that point when starting up again. However, both Adventurer and Explorer modes allow you to also create a separate save point of your own, and load it whenever you want. Access this feature via buttons in the game menu, or use Ctrl-F8/F9 from the main game interface.

Note there is only one slot, so setting a new manual save point will overwrite the previous one. Also this system operates outside Cogmind's regular automatic save/load system, so if you save and exit the game while a run is in progress, that latter save will be the one which is automatically loaded when you return, but you can still always access your same earlier manual save point as long as that same run is in progress.

If no manual saves have been made so far during the current run, a save point will automatically be created at the beginning of each new map, so there is always the option to revert to that point on the current map at any time, or on death. Setting a manual save point will deactivate this feature, lest it override your manually designated save point. A new automated save is not made if more than one-third of your part slots were empty on leaving the previous map.

Using the manual save slot is entirely optional, and runs during which this feature is used are not included for stat uploading or leaderboard purposes. Those who want to play Adventurer/Explorer while avoiding the temptation to use this feature can disable it by setting noManualSaving in /user/advanced.cfg.

Other Options

You can also increase or at least modify the difficulty and content of the experience by activating either challenge modes or special modes, which are described in the later Alternative Rules section. Some special modes in particular make drastic changes to the way the game plays, which might be more your thing.

Core

The most vital component of any robot is its "core". Once a core's integrity is reduced to zero, that robot is destroyed. Cogmind is no exception.

Over time Cogmind's core will evolve, gaining both integrity and the ability to interface with a greater number of parts at once. The reason for this evolution is part of the story to be uncovered.

Inherent Stats

Exposure	100
Visual Range	16
Integrity	250 (+150/evolution)

Energy rate 5/turn
Energy storage 100
Matter storage 300
Heat Dissipation 25 (+3/evolution)

System Corruption

Aside from integrity damage, robots are also prone to system corruption. Any core that is 100% corrupted is destroyed regardless of structural integrity. The most common cause of system corruption is electromagnetic damage.

Cogmind's own system corruption is wiped with each evolution.

Resources

Energy

Once per turn energy is generated by power sources, and is necessary for moving, firing energy-based weapons, and sustaining the operation of some utility systems. Robot cores also generate some energy of their own, though not enough to support significant activity.

Matter

Matter is used to fuse components, and is consumed by ballistic weapons and launchers, which convert it to the appropriate type of ammunition. It can be salvaged from robot remains, though the available amount varies depending on how the robot was destroyed. Cogmind automatically recycles 5 matter from each attached part that is destroyed.

Upkeep

While active, some parts may consume energy (or more rarely, matter) every turn. Utilities and hover/flight units are the most common parts to require upkeep.

Parts

Different robot cores are designed to interface with a unique set of parts. Parts fall into one of four categories: Power, Propulsion, Utilities, and Weapons. The Cogmind is special in that it can dynamically bind and interface with any type of part by expending an amount of matter and energy.

Attaching a part requires 20 energy and 10 matter. Detaching a part expends 10 energy.

Activation

Most parts must be activated in order to use them. Others are permanently "active" (usually when the state has no meaning for them, as with various storage units). Activating/deactivating parts are both free actions (i.e., take no time to perform).

Inventory

Robots can generally carry a few items, but inventory space is very limited unless expanded through utilities. Items contained in inventory do not count against the robot's mass limit for movement speed calculation purposes.

Prototypes

Newly developed more effective components may be found throughout the world, but without testing or prior knowledge it is impossible to identify and distinguish between working versions and their more dangerous faulty counterparts. Faulty prototypes will almost always malfunction and have potentially serious side-effects when attached, and cannot be used or repaired, but may be scanned to obtain the schematic.

Rating

This is essentially a part's "level", and from a player perspective is just a way to quickly compare whether one component is likely more effective than another. However, always consider that prototypes are generally better than common parts with a similar rating, and other factors such as the current situation or a part's particular characteristics may be a reason to favor lower ratings.

Power Sources

Engines, power cores, and reactors supply the power necessary to run other systems. Be aware that most power sources will shut down temporarily when the core begins overheating.

Overloading

Some power sources are capable of overloading, which doubles both energy output and heat generation. However, overloaded power sources also have a chance to cause negative side effects, determined by its "stability" stat.

Propulsion

Treads, legs, wheels, and hover/flight units represent a spectrum from greater mass support and slow speed to lower mass support and fast speed. Exceeding the mass limit provided by active propulsion components will slow movement.

Base speed is measured by the time it takes to move one space, where an "average" speed is 100 time units. Check Cogmind's status screen for an indicator of relative speed as a percentage (100% is average, higher = faster), or see that same page or the main HUD for the actual cost in time.

Context help available for propulsion component stats explains the movement mechanics in more detail.

Note that only one form of propulsion may be active at a time, where different variations of the same form will use the average of their respective stats.

Note that while flying Cogmind can rush directly over blocking robots, even hostile ones, as long as not currently overweight.

Costs

Movement takes some amount of energy, and may produce heat. These amounts are deducted/produced for each move, while power generation and heat dissipation are only applied once per game turn; by extension, if your speed enables you to move many spaces per turn, these costs will be compounded before your other systems can deal with them. Consider this fact when calculating what kind of propulsion you can support with given power sources (and to a lesser extent, heat dissipation utilities).

Upkeep for applicable propulsion components (generally hover and flight units) is only deducted after one absolute game turn has passed.

Overloading

Some hover and flight units are capable of overloading, which boosts performance and associated costs. While overloaded, speed is calculated as if there is twice the number of units active at once, support is increased by 50%, energy costs are doubled, and heat generation is tripled. The performance boost comes at the additional cost of integrity strain (see a propulsion's "burnout" stat). For these parts, the normal activation/deactivation command will instead cycle between off, on, and overloaded states.

Core Movement

Unlike other robots, Cogmind's core itself is capable of movement using a built-in system of an unknown nature. It is automatically engaged when no other forms of propulsion are active, and while inefficient when weighed down, it is otherwise relatively fast. Cogmind's core moving under its own propulsion has the following stats:

Time/Move	50
Energy	1
Heat	0
Support	3
Penalty	50

Utilities

The largest and most versatile category of parts is made up of those classified as some form of utility. Each utility is built around a single capability, the effect of which is described on the part's info page. Where applicable, the description also indicates whether the effect stacks, an important factor for mechanics purposes.

Devices

The broadest subcategory of utilities provides a wide range of unique functions.

Processors

Lightweight components that generally serve to augment an existing ability. Once attached, a processor cannot be removed without rendering it useless.

Hackware

Processors dedicated to hacking are categorized separately for easy recognition. These provide either offensive or defensive capabilities applicable to machine and robot hacking. Once attached, hackware cannot be removed without rendering it useless.

Storage

A relatively small utility subcategory purely for holding resources or other components. Normally these are used to expand capacity while attached, but in the case of energy and matter storage they can also be detached with their contents intact. Non-attached storage components show their current contents in parenthesis. Surplus collected matter and generated energy will also be automatically transferred to non-full storage units in inventory, but cannot be accessed for use unless those parts are attached.

Armor

Protective gear, some of which does not indicate a particular effect, instead using high coverage and integrity to reduce the chance that other parts will be damaged. Unlike other parts, armor can never be instantly destroyed by critical strikes, taking 20% more damage instead.

Weapons

See weapon stat context help for explanations of weapon-specific mechanics.

Guns

Average-strength weapons that inflict one of kinetic, thermal, or electromagnetic damage.

Cannons

The most powerful method of striking a target with concentrated damage, but heavier than guns, and costlier to support.

Launchers

Area effect weapons that generally inflict explosive or electromagnetic damage, though the amount of damage decreases with distance from ground zero.

Special Weapons

Tools and other non-conventional weapons.

Melee Weapons

Limited to close range combat, but capable of inflicting one of impact, slashing, or piercing damage, each of which has a unique effect.

Multiple Projectiles

Some weapons indicate that they shoot multiple projectiles. In such cases, the projectile stats displayed apply to each projectile fired, e.g. a damage value listed as "9-12" for a dual-projectile shotgun can do up to 18-24 damage, although note that the hit roll for projectiles are also calculated individually, meaning that one of the projectiles may miss entirely, or each projectile could hit a different location on the target, thereby spreading the damage.

Overloading

Some energy weapons are capable of overloading, which doubles damage and energy cost while

generating triple the heat. Overloaded projectile heat transfer is also one level higher than usual where applicable. However, firing an overloaded weapon has a chance to cause negative side effects, as reflected in that weapon's "stability" stat.

Time

Turns and the time required to perform actions are measured in time units. One absolute turn is equivalent to 100 units of time. All robots on the map are placed in a queue ordered by the amount of time they've spent so far, with the robot at the front of the queue always acting first. On performing an action, that robot is moved in the queue to a new position based on its total amount of action time relative to all other robots. Their new position might still be at the front if it was an extremely quick action, or very far back in the queue due to a costly action, in which case it will be longer before that robot can act again.

Action Costs:

Pick Up	100
Attach	100
Pick Up+Attach	150
Detach	50
Swap	150
Drop	50
Fire	200
Ram	100+
Rewire	100
Wait	100
Move	Varies by propulsion and mass

Firing more than one weapon in a single volley is more time-efficient, and that efficiency increases the more weapons you fire at once:

#Weapons	Total Time
2	300
3	325
4	350
5	375
6	400

As you can see, firing a huge volley has its advantages. You can fire six times as many weapons for only twice the cost, and every weapon beyond the sixth incurs no additional cost. Note that regardless of individual or cumulative weapon delay modifiers, the time cost of a volley can never be reduced below 25.

Combat

Volley

When attacking, all active weapons in range of the target are fired at once. Firing a single weapon takes twice as long as a standard action, and subsequent weapons fired in the same volley will take less and less extra time, hence firing more weapons at once is always more time efficient. See the HUD's volley analysis readout to check the actual time required to fire the currently active weapons (which may be further modified by individual weapon delays), as well as the total resource cost of the volley.

Although the full resource cost of the intended volley must be available before firing, energy and matter costs for each weapon are not deducted until that weapon actually fires. All weapons immediately begin generating heat as soon as the volley begins, but for volleys that will require two or more turns, the heat produced is averaged over the total number of turns required to fire.

Targeting

Targeting is actually tested on a finer grid than is visible onscreen. Each map space is divided into a 9x9 grid of squares, and as robot sizes vary (S/M/L), they may take up more or fewer of these squares. This has several implications, e.g. you may be able to hit a larger target before it has completely rounded a corner, while smaller targets may require a more direct line-of-sight. It also means that smaller targets are actually easier to shoot around than larger ones. As long as the targeting line is green Cogmind has a clear LOF to the target, even if it looks like it passes through another robot or obstacle.

Hit Chance

Many factors affect the chance to hit a target. The chance shown in on-map popups and the scan readout take into account all factors except those which affect individual weapons in the volley. Per-weapon chances are instead shown in the parts list next to their respective weapons once firing mode is activated and the cursor is highlighting a target.

Base hit chance before any modification is 60%.

Volley Modifiers:
- +3%/cell if range < 6
- +20%/30% if attacker in standard/high siege mode (non-melee only)
- +attacker utility bonuses
- +10% if attacker didn’t move for the last 2 actions
- +3% of defender heat (if heat positive)
- +10%/+30% for large/huge targets
- +10% if defender immobile
- +5% w/robot analysis data
- -1~15% if defender moved last action, where faster = harder to hit
- -5~15% if defender running on legs (not overweight)
  (5% evasion for each level of momentum)
- -10% if attacker moved last action (ignored in melee combat)
- -5~15% if attacker running on legs (ranged attacks only)
  (5% for each level of momentum)
- -3% of attacker heat (if heat positive)
- -10%/-30% for small/tiny targets
- -10%/-5% if target is flying/hovering (and not overweight or in stasis)
- -20% for each robot obstructing line of fire
- -5% against Cogmind by robots for which have analysis data
- -defender utility bonuses

Weapon-specific Modifiers:
- +utility bonuses
- -recoil (from other weapons)

After all modifiers are applied, regardless of the final value hit chance is capped at 95% (or 100% for melee attacks). Similarly, there is always at least a 10% chance to hit a target. The cap is not applied to the per-weapon chances shown in the parts list in order to let you know how much surplus hit chance is available should tactical changes occur.

Sometimes the log will claim you "missed" a target, then destroyed it. That’s because you got lucky and

the random miss trajectory still hit the target! This is more likely to happen with closer, larger targets as per the mechanics described in the targeting section above.

Note that anything caught in an explosion radius is always hit--explosions automatically damage everything in each space they traverse, be it walls, items, or robots.

Volley modifiers reordered by type for comparative reference:
* +3%/cell if range < 6

* +10%/+30% for large/huge targets
* -10%/-30% for small/tiny targets

* +3% of defender heat (if heat positive)
* -3% of attacker heat (if heat positive)

* +20%/30% if attacker in standard/high siege mode (non-melee only)
* +attacker utility bonuses
* -defender utility bonuses

* +10% if defender immobile
* +10% if attacker didn't move for the last 2 actions
* -10% if attacker moved last action (ignored in melee combat)
* -1~15% if defender moved last action, where faster = harder to hit
* -10%/-5% if target is flying/hovering (and not overweight or in stasis)
* -5~15% if defender running on legs (not overweight)
    (5% evasion for each level of momentum)
* -5~15% if attacker running on legs (ranged attacks only)
    (5% for each level of momentum)
* -20% for each robot obstructing line of fire

* +5% w/robot analysis data
* -5% against Cogmind by robots for which have analysis data

Guided Weapons

Guided weapon behavior and operation are different from other weapons. Their projectiles follow a set path designated by a number of waypoints, and will always hit their target, but may not be fired as part of a volley. While a guided weapon is active, only that weapon will fire. Left-click or press Enter to set each waypoint, and use right-clicks or Escape to cancel a waypoint. When ready to fire, put the cursor over the final target and press 'f'. Players preferring pure mouse control and attempting to target an empty space or space out of view can also click on the last waypoint to fire, or assign all waypoints up to the weapon's limit then click on a final target direction.

Coverage/Exposure

Each part has a "coverage" rating which determines its likeliness to be hit by an incoming attack. Values are relative, so attacks are weighted towards hitting parts with higher coverage. Robot cores also have their own "exposure" rating which determines their likeliness to be hit; this value is considered along with part coverage when determining whether an attack will strike the core. The exact chance of a core/part to be hit is shown in parenthesis after its exposure/coverage on the relevant info screen. You can also press 'c' to have the main HUD's parts list display a visualization of relative coverage, i.e. longer bars represent a greater chance for a given part to be hit.

Some examples of how coverage determines hit locations will help understand how that stat actually works.

Example 1: Cogmind's core has an exposure of 100. Say you equip only one part, a weapon which also has a coverage of 100. Their total value is 200 (100+100), so if you are hit by a projectile, each one has a 50% (100/200) chance to be hit.

Example 2: You have the following parts attached:
  Ion Engine (60)
  Light Treads (120)
  Light Treads (120)
  Medium Laser (60)
  Assault Rifle (100)
With your core (100), the total is 560, so the chance to hit each location is:
  Ion Engine: 60/560=10.7%
  Light Treads: 120/560=21.4% (each)
  Medium Laser: 60/560=10.7%
  Assault Rifle: 100/560=17.9%
  Core: 100/560=17.9%

Enemy robots work the same way, so the more parts you blow off, the more likely you are to hit and destroy their core. Armor plating has a very high coverage, so it's more likely to be hit, while tiny utilities such as embedded processors have very low coverage, so you can expect them to last much longer (unless you have little or nothing else covering you). As you progress, your core will become more and more protected by attached parts, because you'll have many more of them, but by that time there are other dangers such as system corruption.

Damage Overflow

When a part is destroyed by damage that exceeds its remaining integrity, the surplus damage is then applied directly to the core or another part as chosen by standard coverage/exposure rules. No additional defenses are applied against that damage. Exceptions: There is no damage overflow if the destroyed part itself is armor, and overflow damage always targets armor first if there is any. Critical strikes that outright destroy a part can still cause overflow if their original damage amount exceeded the part's integrity anyway. Damage overflow is caused by all weapons except those of the "gun" type, and can overflow through multiple destroyed parts if there is sufficient damage.

Secondary Targeting

If a volley including only "gun"-type weapons destroys a target robot, any and all remaining guns in that volley will lock onto another target, prioritizing the most easily hit armed and active hostile currently in range. This mechanic is also called "gunslinging," although certain capabilities may be acquired to expand this mechanic beyond guns alone.

Salvage

How much of a robot remains to salvage when it is destroyed depends on the value of its cumulative "salvage modifier" which reflects everything that happened to it before that point. This internal value is initially set to zero, and each projectile that impacts the robot will contribute its own weapon-based salvage modifier to the total. Some weapons lower the value (most notably ballistic cannons), others have no meaningful effect on it (most guns), while certain types may even raise it, ultimately increasing the likelihood of retrieving useful salvage.

When a robot is destroyed, it leaves an amount of matter equivalent to its salvage potential (usually a random range you can see on its info page), modified directly by the salvage modifier. This means a large enough negative salvage modifier, e.g. from explosives or repeated cannon hits, has the potential to reduce the amount of salvageable matter to zero. A positive salvage modifier can never increase the

resulting matter by more than the upper limit of a robot's salvage potential.

The chance for the robot's parts to survive, checked individually for each part, is ([percent_remaining_integrity / 2] + [salvage_modifier]), thus more damaged parts are more likely to be destroyed completely along with the robot. But even if that check succeeds, there are still two more possible factors that may prevent a given part from dropping. If the robot has any residual heat, parts can be melted, the chance of which is ([heat - max_integrity] / 4), so again less likely to affect large parts (though still possible, especially at very high heat levels). If the robot was corrupted, parts can be "fried," the chance of which is [system_corruption - max_integrity]; by the numbers, this will generally only affect small electronic components like processors, and sometimes devices. Each salvageable part left by a corrupted robot also has a corruption% chance to itself be corrupted, specifically by a random amount from 1 to (10*[corruption]/100), and when attached will increase Cogmind's system corruption by the same value as well as possibly cause another side effect.

Note that robots built by a fabricator do not leave salvageable parts. Also, anything in a robot's inventory (not attached) is always dropped to the ground, regardless of salvage modifiers or other factors.

Spotting

Even when you can see a hostile robot, they will not always notice you immediately. They can only register a target on their own turn, so if you are moving very quickly you can potentially pass into and beyond their field of view before they even have time to react. Sometimes when quickly rounding a corner and spotting an enemy, there is still time to duck back without being seen. That said, if you pass in and out of a hostile robot's field of view before they react, they may still take some notice and decide to investigate whatever they think they saw. Different AIs have a different chance to take notice, with those robots designed for surveillance, for example, being much more likely to investigate a potential passing target.

Another possibility to be aware of: while your own sight range is normally similar to that of other robots (approximately 16 spaces), if you enhance your visual sensors you can spot robots from afar before entering their field of vision. Likewise, hostiles with their own sensors may spot you before you can see them.

To know whether a robot has actually spotted you or not, there are two useful indicators. The first, directly on the map itself, is the sudden glow of their background when they spot you for the first time (also may display as a flashing '!' depending on other UI settings). After that initial sighting you can still refer to the scan window, which displays a red exclamation mark for any hostile robots aware of your presence.

Note that sight range is also reduced when line of sight passes through machines, making it possible in certain environments to get closer to hostiles before being spotted. Cloaking utilities also reduce the range at which enemies can spot you.

Attack Resolution

With so many mechanics playing into an attack, especially later in the game when there are numerous active parts involved at once, min-maxers looking to optimize their build may want a clearer understanding of the order in which all elements of an attack are carried out. This manual already covered hit chances above, but there are many more steps to what happens once an attack hits a robot. Below is an ordered list detailing that entire process, which applies to both ranged and melee combat. Note that you most likely DO NOT need to know this stuff, but it may help answer a few specific questions min-maxers have about prioritization.

1. Check if the attack is a non-damaging special case such as Datajacks, Stasis Beams, Tearclaws, etc., and handle that before quitting early.

2. Calculate base damage, a random value selected from the weapon's damage range and multiplied by applicable modifiers for overloading, momentum, and melee sneak attacks, in that order. Potential damage range modified by Melee Analysis Suites, Kinecellerators, and Force Boosters here.

3. Apply robot analysis damage modifier, if applicable (+10%).

4. Apply link_complan hack damage modifier, if applicable (+25%).

5. Apply Particle Charger damage modifier, if applicable.

6. Reduce damage by resistances.

7. Apply salvage modifiers from the weapon and any Salvage Targeting Computers.

8. Determine whether the attack caused a critical hit.

9. Split damage into a random number of chunks if an explosion, usually 1~3. The process from here is repeated for each chunk.

10. Store the current damage value as [originalDamage] for later.

11. Apply the first and only first defense applicable from the following list: phase wall, 75% personal shield (VFP etc), Force Field, Shield Generator, stasis bubble, active Stasis Trap, Remote Shield, 50% remote shield (Energy Mantle etc.), Hardlight Generator.

12. Store the current damage value as [damageForHeatTransfer] for later.

13. Choose target part (or core) based on coverage, where an Armor Integrity Analyzer first applies its chance to bypass all armor, if applicable, then available Core Analyzers increase core exposure, before finally testing individual target chances normally.

14. Cancel critical strike intent if that target has applicable part shielding, or if not applicable to target.

15. If targeting core, apply damage and, if robot survives, check for core disruption if applicable.

16. If targeting a power source with an EM weapon, check for a chain reaction due to spectrum.

17. If not a core hit or chain reaction, prepare to apply damage to target part. If the part is Phase Armor or have an active Phase Redirector, first reduce the damage by their effect(s) and store the amount of reduction as [transferredCoreDamage] for later, then if part is Powered Armor reduce damage in exchange for energy (if available), or if part is treads currently in siege mode reduce damage appropriately. If part not destroyed then also check for EM disruption if applicable.

18. If part destroyed by damage, record any excess damage as overflow. If outright destroyed by a critical hit but damage exceeded part's integrity anyway, any excess damage is still recorded as overflow.

19. If part was not armor and the attack was via cannon, launcher, or melee, transfer remaining damage to another random target (forcing to a random armor if any exists). Continue transferring through additional parts if destroyed (and didn't hit armor).

20. If part not destroyed, check whether heat transfer melts it instead.

21. Apply [transferredCoreDamage] directly to the core if applicable, with no further modifiers.

22. Apply damage type side effects:

* Thermal weapons attempt to transfer ([damageForHeatTransfer] / [originalDamage])% of the maximum heat transfer rating, and also check for possible robot meltdown (the effect of which might be delayed until the robot's next turn).

* Kinetic damage may cause knockback.

* The amount of EM corruption is based on [originalDamage].

* Impact damage may cause knockback, and applies corruption for each part destroyed.

Melee Combat

Melee weapons can only hit adjacent targets, and can only be used one at a time. Activate a melee weapon and bump (move) into a target to attack it. Other options are to left-click on the target itself, or in that target's general direction. You can also force a melee attack, generally only necessary to use a melee weapon on terrain or a machine, by holding Ctrl-Shift while moving in a certain direction. (Note to vi-key users: Forcing a melee attack this way using vi-keys is not possible because the commands overlap with part swapping, use 'f' to enter targeting mode instead.)

Hit Chance

Melee attacks use the same hit chance calculations as ranged combat, with a few exceptions:
- Base hit% is 70
- No range modifier
- No heat modifiers
- Utility modifiers use those applicable to melee combat

Momentum

All melee weapons benefit from greater momentum in the direction of the attack. Moving multiple consecutive spaces in a row prior to an attack adds additional momentum, up to a maximum of 3. Current momentum is displayed as a number at the end of the HUD's movement data readout.

The damage bonus on a successful melee hit is +1~40% for non-piercing weapons, or +2~80% for piercing weapons, calculated as: ([momentum] * [speed%] / 1200) * 40). For piercing attacks the multiplier is 80 instead of 40. Speed% is current speed as a percentage of average speed, 100. Thus the more momentum and speed with which you attack, the greater the damage. Your status page shows the damage bonus taking current momentum into account. Note that while some utilities help increase momentum, the maximum bonus damage for a given weapon type cannot be exceeded.

Stopping, performing a non-movement action, or otherwise moving in a direction that does not match the current momentum resets your momentum to 0, except when moving diagonally in the same general direction (e.g. turning from southeast to south). The latter case instead reduces total momentum by 1. Also note that technically any melee attack can take advantage of previously accumulated momentum, regardless of direction. For example, approaching the southern side of a target from the east and attacking northward while passing below it will apply whatever momentum value is displayed in the HUD.

Sneak Attacks

Melee attacking an enemy that has not yet noticed you gives a base hit chance of 120%, and a +100% damage bonus which stacks with any momentum bonus. Sneak attacks work on any non-fleeing neutral

targets as well, since they don’t expect you’ll suddenly attack them!

Multi-Wielding

Although only one primary melee weapon can be active at a time, other attached but inactive melee weapons have a chance to carry out "follow-up attacks" alongside the primary weapon. In Tactical HUD mode, that chance is shown next to each applicable melee weapon. Follow-up attacks are more likely when the primary weapon is slower than a given backup melee weapon. The chance may also be affected by supporting utilities.

Multiple follow-up attacks by different melee weapons are possible in the same action. Each weapon is checked separately, and once confirmed any follow-up attack has a +10% to hit the target. Each weapon incurs no additional time cost to attack aside from modifying the total attack time by half of their time delay (whether positive or negative). Momentum bonuses that apply to the primary weapon all apply to follow-up attacks as well. The benefits of all actuators also apply to every follow-up attack. If the target is destroyed by an earlier attack, any remaining follow-up attacks will switch to another target in range if there is one.

Datajacks cannot be involved in follow-up attacks, and these attacks are only applicable against robot targets.

Ramming

As a last resort, Cogmind can ram other robots to damage and/or push them out of the way. Damage is a random amount from 0 to (((10 + [mass]) / 5) + 1) * ([speed%] / 100) * [momentum], where speed% is current speed as a percentage of average speed (100) and effective momentum is a combination of both Cogmind and the target’s momentum. However, the damage per attack is capped at 100 before the roll. Smashing into a robot headed straight for you can deal some serious damage, though there are serious negative consequences to go with this unarmed attack, and half as much damage is inflicted on Cogmind as well. Ramming enables the collection of a small random amount of matter per collision.

Ramming with active treads or legs always avoids self-damage and destabilization. Treads have a per-tread chance to instantly crush targets of medium size and below which have no more than 50 remaining core integrity. Crushed robots have their salvage modified by -20. Legs have a 20% chance per leg to kick the target out of the way. (Not applicable against huge targets.)

The time cost to ram is the greater (slower) of 100 and your current move speed.

It is also possible to ram non-robot targets including walls and machines by using the "force melee attack" command (Ctrl-Shift + Movement Keys, or Ctrl-Shift+LMB) without any active melee weapons. Ramming of this type uses the same damage formula as above, but the results are always maximized, and therefore you can know in advance whether the damage will be sufficient to destroy the target. The expected result will be indicated via the usual collision warning when you attempt to ram. Note that ramming a solid object is much more dangerous than ramming a robot, and Cogmind not only takes the same amount of damage that is inflicted on the target, but also suffers other negative side effects as well. The type of propulsion does not affect the results, beyond the indirect effects of carrying capacity and speed. To repeatedly force melee by simply moving into a valid target (or make such an attack compatible with vi-keys), you can toggle force melee mode via the Special Commands interface (Spacebar) or Shift-Alt-e.

Damage Types

Weapon damage is classified into 7 primary types, each with unique properties and effects. Other less common types may be discovered on your travels.

Thermal

Thermal energy weapons generally have a shorter range, but benefit from a more easily predictable damage potential and little or no recoil. Thermal damage also generally transfers heat to the target, and may cause meltdowns in hot enough targets.

Kinetic

Ballistic weapons generally have a longer effective range and more destructive critical strikes, but suffer from less predictable damage and high recoil. Some kinetic projectiles, especially hypervelocity variants, are capable of penetrating one or more consecutive targets. Kinetic cannon hits also blast usable matter off target robots and have a damage-equivalent chance to cause knockback, with a (10 - range) * 5 modifier and a +/-10% per size class (T/S/M/L/H) difference between the target and medium size (targets knocked into another robot may also damage and displace it, see Impact below). The amount of matter created per cannon projectile impact is randomized from between zero up to the absolute value of the weapon's salvage modifier, but only applies for those weapons with a negative salvage modifier below -2.

Electromagnetic

Electromagnetic (EM) weapons have less of an impact on integrity, but are capable of corrupting a target's computer systems. Anywhere from 50 to 150% of damage done is also applied as system corruption. (Cogmind is less susceptible to EM-caused corruption, but still has a damage% chance to suffer 1 point of system corruption per hit.) EM-based explosions only deal half damage to inactive items lying on the ground, but can also corrupt them.

Some EM weapons cause "disruption", which is a per-shot chance to temporarily disable an active part on impact. If a robot core is struck, there is half this chance the entire robot may be disabled. While disrupted a robot may be rewired by bumping into it (see Hacking Robots).

Explosive

While powerful, explosives generally spread damage across each target in the area of effect, dividing damage into separate chunks before affecting a robot, where each chunk selects its own target part (though they may overlap).

Explosions also tend to reduce the amount of salvage remaining after destroying a target, and sometimes cause cave-ins in non-reinforced areas.

Impact

Impact melee weapons have a damage-equivalent chance to cause knockback, with a +/-10% per size class (T/S/M/L/H) difference between attacker and target. Targets knocked into another robot may also damage and displace it. A robot hit by another displaced robot has a chance to itself be displaced and sustain damage, where the chance equals the original knockback chance further modified by +/-10% per size class difference between the blocking robot and the knocked back robot, and the resulting damage equals [originalDamage] (see Attack Resolution), further divided by the blocker size class if that class is greater than 1 (where Medium = 2, and so on).

Impact weapons ignore relative coverage when determining which components to affect, and are therefore effective at destroying fragile systems regardless of armor and other protection. By ignoring coverage, there is an equal chance for an attack to hit any given part, where each slot of a multislot part

also has a separate and equivalent chance to be hit. A target's core also counts as one slot for that purpose. In addition, for every component crushed by an impact, its owner's system is significantly corrupted (+25-150%), though electromagnetic resistance can help mitigate this effect. (Cogmind is less susceptible to corruption caused in this manner.)

Slashing

Slashing melee weapons are generally very damaging, and most are also capable of severing components from a target without destroying them.

Piercing

Piercing melee weapons inflict less collateral damage, but are more likely to hit a robot's core, and get double the melee momentum damage bonus.

Heat

Movement, firing weapons, and running power sources and some utilities generates heat. Some of this heat is naturally dissipated by the core's own heat sinks, but not enough to deal with heat generated by numerous/large energy weapons. Heat sinks and cooling systems can be used to avoid overheating, which can have a wide range of negative effects. Heat is only a significant issue for robots that rely heavily on energy weapons. However, note that when firing a volley the heat produced is averaged over the volley's turn duration rather than being immediately applied all at once.

Side Effects

Once heat reaches the "Hot" level (200+), active utilities and weapons may be temporarily disabled. At "Warning" levels (300+) power sources are likely to shut down. Many more serious (and permanent) effects are possible, especially at higher heat levels.

Disabled power sources automatically restart when possible, but other parts must be manually reactivated.

Heat effects are not calculated until after the dissipation phase, so heat can temporarily spike very high with no side effects as long as there are sufficient utilities to dissipate it.

Hacking Machines

Interactive machines appear in color rather than gray. Access their interface by bumping (moving) into the solid color space containing the representative letter.

Factors

Each function/command at a machine has its own base difficulty to hack, though there are numerous other factors at play in determining your chance of success.

* Security Level: Each class of machines exists in three different varieties, rated by security level from 1 to 3. Higher security machines provide greater functionality, but are correspondingly more difficult to hack.

* Utilities: Active hackware contributes to hacking attempts, most directly by making each hack easier.

* System Corruption: Being corrupted decreases the chance of success (-corruption/3%).

* Operators: Having a network of operators under your control helps bypass machine security systems, though the benefits decrease with each additional operator.

Process

From your point of view, hacking is simply connecting to a system and choosing/entering commands. However, the system itself responds in three phases:

* Detection: Initially your presence is unknown, but with each subsequent action there is a chance your activity will be detected. Detection happens more quickly at higher security machines, and becomes more likely with each failed hack, but can be mitigated by defensive hackware. Accessing the same machine more than once also increases the chance of detection, if it was previously hacked but not traced.

* Tracing: As soon as suspicious activity is detected, the system attempts to locate it. Failing hacks increases the speed of the trace, more quickly for worse failures. If a session is terminated while a trace is in progress, that trace will resume from where it left off if a connection is reestablished.

* Feedback: Once traced, the system is permanently locked and may attempt to counterattack the source of the unauthorized access, which either causes system corruption or disables connected hackware.

Note that successful hacks (especially those which were difficult to pull off) have a chance to cause an increase in local alert level, though this result is less likely while using defensive hackware.

Terminals

Terminals are the most common type of interactive machine, providing access to local and central networks and databases. At the time of access, the terminal interface lists all potential hacking targets found for direct hacking. Unlisted targets can be hacked by manually entering the associated command, though so-called "indirect hacking" is more difficult (-15% chance per security level, though offensive hackware can be used to counteract this difficulty). The precise difficulty of indirect hacks is also an unknown factor, making them more dangerous for the inexperienced hacker.

Terminal records are purely informational, and will teach you more about the world and its inhabitants. Those you have never hacked before, and therefore do not yet appear in your lore collection, will have a ’!’ prepended to their target name. This reminder can be deactivated in /user/advanced.cfg via the markUndiscoveredLore option.

Operators are stationed at some terminals. Destroying the operator and retrieving its data core will extract its dynamic key and provide a 1.5x bonus to the first hack at that operator’s terminal, but only if the connection is established before the key expires. Nearby trap locations can also be de-encrypted from pre-expiry data cores.

High-security terminals have direct access to more functions at once.

Fabricators

Fabricators construct parts and robots based on schematics. Schematics can be acquired by using scanalyzers or hacking terminals, then loaded into a fabricator which will report the resources required to complete construction. Matter is drawn from the local fabrication network.

High-security fabricators are capable of faster construction.

Repair Stations

Repairing parts is a two-stage process. First instruct the station to scan a component currently in your inventory. Then initiate the repair process, which both fixes broken parts and restores them to full integrity.

The separate "refit" command scans a robot and attempts to restore missing functionality using simple backup parts.

High-security repair stations are capable of faster repairs.

Recycling Units

Recycle components by loading them from your inventory and initiating the process. Recycling units collect matter until they reach a certain quota (500), after which it is transferred away to a central system. Instruct the unit to report matter to examine its current stores. Hack the retrieve matter command to have it eject its contents.

Similarly, unprocessed parts can be listed and retrieved via other system commands.

Scanalyzers

Scanalyzers provide schematics for existing parts. Insert a part from your inventory, followed by the instruction to analyze. See your status page for access to a list of schematics currently in your possession, and note that in part info visualization mode (’q’) data preceded by a ’+’ indicates you have a schematic for that part.

More powerful scanalyzers generally require fewer scans to successfully create a proper schematic, and are less likely to damage the scanned part.

Manual Hacking

Manual entry of hacking commands has multiple uses. At terminals this feature can be used to indirectly hack unlisted targets. The text to enter commands is learned by observing the results window output during a regular listed hack. (For example, enter "Alert(Check)" at any terminal to attempt that hack.)

To simplify repeat hacks, all previously entered manual commands are stored in a buffer; once the >> command prompt is active, press Up/Down arrows to cycle through them. (If necessary you can edit this buffer directly, found in /user/buffer.txt, though editing should not be done while the game is running.)

Manual hacking also supports autocompletion. As you type it will compile a list of all matching commands, showing the first as grayed out text. Press tab to accept the gray text rather than typing it out in full. For subcommands, those parts of a command after a left parenthesis, if there are multiple options a list will appear allowing you to optionally press up/down to cycle through them and tab to accept. Continuing to type will filter the list down further. From the outset all common commands are considered for the autocompletion list, while unauthorized hacks you learn on your travels are considered only after you discover them (or if you have entered them directly at least once before).

Successful indirect hacking of central "database-related" targets (queries, schematics, analysis, prototypes) incurs a 25% chance to trigger a database lockout, preventing indirect access to those types of targets at every terminal on the same map. The chance of a lockout is reduced while using defensive hacking utilities.

Specific to manually hacking robot schematics and analyses, note that entering the class name will

auto-select the best relevant variant available at the given terminal, from among those you don't already have. (For example, "Schematic(Grunt)" tries to hack the best applicable Grunt-type variant.) Manually hacking robot schematics/analyses and loading schematics also allows for case-insensitive partial string matching. For example the string "troop" would be automatically expanded to "G-47 Trooper" because that is the first substring match. If more than one match exists, the first one found in the database is always chosen.

Manual hacking can also be used to enter unique codes obtained on your travels. Any such codes learned in the current run will be listed automatically on beginning a manual hack. Non-mouse players can type the code prefix "\\" to cause the focus to switch to that menu in order to select an option by letter (or Escape/Backspace to return to the normal text entry). If the focus is already in the menu, pressing Escape returns to the text area to allow manual entry or editing as necessary.

Manual hacking commands are case-insensitive, including both the command itself and any optional arguments.

Hacking Robots

Access a robot's system using either a melee or remote datajack. While the parse_system hack can be used on almost any robot, most other hacks are limited to certain targets. Using a datajack on a target is not considered a hostile action, and targets won't even notice.

Requirements

Any hacks listed in the Basic category are accessible with nothing more than a datajack. "RIF" category hacks require that you've installed a Relay Interface Framework via the dedicated machine found inside any garrison. Most hacks require having also attached a robot-appropriate Relay Coupler, which can be acquired in a number of ways.

Cost

The cost of performing a coupler-based hack is listed next to the hack, and deducted from the coupler's remaining value when performed.

Manual Hacking

Manual entry of standard robot hacking commands is not necessary, but offered as an optional method.

To simplify repeat hacks, all previously entered manual commands are stored in a buffer; once the >> command prompt is active, press Up/Down arrows to cycle through them. (If necessary you can edit this buffer directly, found in /user/buffer_robot.txt, though editing should not be done while the game is running.)

Manual hacking also supports autocompletion. As you type it will compile a list of all matching commands, showing the first as grayed out text, possibly followed by a number indicator if there is more than one match available. Press spacebar or tab to accept the gray text rather than typing it out in full, or press up/down to cycle through other options.

Manual hacking can also be used to enter unique codes obtained on your travels. Any such codes learned in the current run will be listed automatically on beginning a manual hack. Non-mouse players can type the code prefix "\\" to cause the focus to switch to that menu in order to select an option by letter (or Escape/Backspace to return to the normal text entry).

Manual hacking commands are case-insensitive, including both the command itself and any optional arguments.

Rewiring

While a robot is disrupted (usually after taking some forms of EM damage to its core), bump into it to attempt a rewire, which if successful will assimilate it. No additional utilities are required, though the chance of success is improved by offensive hackware. Disrupted allies can be instantly reactivated in the same manner.

The base chance of rewiring success is 10%, or 50% with a datajack.

If flying, note that you'll need to switch to some other form of propulsion in order to come down and rewire a target via bumping.

Allies

At various points you may find help in the form of friendly robots that will assist you, or even follow your orders.

Types

Friendly robots can be divided into three categories.

* Drones: This closest type of ally transmits everything it sees as it explores new areas, serving as a second source of visual information.

* Servants: Robots under your direct control will follow you and accept orders.

* Allies: Some robots will not take orders from you, but share your alignment and will fight a common enemy. These allies may or may not follow you, depending on the circumstances.

Orders

Orders can be issued to any robots listed in the allies menu (F6), which includes only those controllable allies currently in sight. (Orders can also be issued directly on the map: Shift-right click on a robot, or press 'o' then its corresponding number, or enter keyboard look mode, move the cursor over the robot and press 'o'.)

By default these robots will follow you and, in the case of combat-capable robots, attack on their own initiative. Both general and specific orders are available:

* STAY: Remain at the current position, moving aside only to avoid blocking other robots. For combat robots this is equivalent to "defend", as they will still chase down attackers, and return to the same position once any enemies have been defeated.

* ROAM: Wander aimlessly. For combat-capable robots, this is the equivalent of "hunt", because they will track down and engage hostiles.

* FOLLOW: Follow you.

* GUARD: As FOLLOW, but follows the specified robot and prioritizes attacking any robot that is attacking the guarded robot.

* AID: As FOLLOW, but follows the specified robot and prioritizes attacking any robot that the aided robot attacks. Non-combat robots may have other class-specific behavior for AID.

* TUNNEL: Dig a tunnel from current position to target.

* DROP: Drop all inventory items, then ROAM within a small area.

* PICKUP: Pick up any items within your current field of vision, starting with those closest to you, then FOLLOW you once inventory is full or there are no more visible items.

* COLLECT: ROAM and collect any items found, then FOLLOW you once inventory is full.

* EXPLORE: Visit unexplored areas of the map to expand known area.

* RETURN: A drone-specific command that instructs them to return to your drone bay. If no bay is available, the drone will simply follow you.

Benefits

A number of allied robots provide unique benefits when under your control. Most of these will be obvious in the right circumstances, but a couple may be difficult or impossible to discover and are therefore recorded here:

* Operators: When hacking machines, one allied Operator confers a +10% to the chance of success, while the benefit of each additional operator is halved (+5%, +2%, +1%), with each operator providing no less than +1%. Operators will also identify hidden doors and traps within your field of vision, and report floor-wide changes in security level. All of the above benefits are only available from those operators within 20 spaces of you (by direct range, line of sight not required).

* Mechanic: When instructed to AID an ally they enter repair mode, in which they attempt to repair (in priority order:) that ally, or Cogmind, or the closest ally in need of repairs. Repair prioritizes robot cores, broken parts, restoration of missing functionality, followed by part integrity. Field repairs on the latter are only capable of restoring a part up to 50% integrity. Repairs cannot be made while either the target is in combat or the Mechanic itself is taking fire. Mechanics are not capable of repairing Cogmind's core. For keyboard users there is a shortcut to simultaneously order all visible controllable Mechanics to AID, by pressing 'r' once in order mode.

Intel

Map intel is accessible via the top-center display (F7). Intel includes the locations of machines, squads, stockpiles, and sabotage results. Gain intel by hacking terminals, which adds glowing markers to the map at their respective locations. Squad markers only indicate their last reported location at the time of hacking, so that information will be outdated before long and those markers therefore disappear on reaching that area. For convenience, all previously seen machines are permanently added into the intel database (because they are stationary).

Click on an intel entry (or enter map intel mode with 'm') to toggle the filter setting for each type of intel.

Traps

Traps are most often found in groups called arrays, and appear in a variety of layouts including rows, blocks or sparser "minefields". They are, however, usually triggered individually.

Triggering

Moving over a trap does not always trigger it. The chance is instead determined by the form of propulsion:

Treads	100%
Legs	75%
Wheels	50%
Hover/Core	40%
Flight	20%

If overweight, the chance is always 100%, regardless of propulsion type.

Once triggered, the trap at that position is spent and will not trigger again. (Some traps exhibit special behavior in this regard.)

Note there are also multiple undocumented circumstances under which a trap may be triggered, not simply regular movement. Also, stasis traps are a special case and their chance to trigger is the opposite of the values listed above (Flight ex: 100 - 20 = 80%).

Press ’>’ while on top of a known trap (or left-click on self) to force it to trigger.

Detection

Cogmind has a 1% chance to detect each trap in view, checked every turn, and boosted by any active Trap Scanners. Hack terminals to reveal entire trap arrays at once. Nearby allied operators will also reveal hidden traps within your field of vision, and local trap locations can also be extracted from a pre-expired operator data core.

Removal

The simplest way to remove one or more traps is to destroy the section of floor housing it. Arrays of traps can be permanently disarmed via terminal hacking, and disabling traps on an individual basis is a possible side effect of hacking one with an active melee datajack.

Reprogramming

By default traps are programmed to trigger on detection of any foreign robot, a setting that can be modified via hacking. Known traps may be reprogrammed with an active melee datajack by bumping them, and entire arrays of traps can be reprogrammed at once from nearby terminals. Reprogrammed traps will no longer recognize Cogmind or allies as a threat, instead reacting to local combat robots. As part of the new programming, each trap’s sensitivity will also be set dangerously high, ensuring it will trigger whenever a hostile moves over it, regardless of propulsion.

Alternative Rules

Over the years I’ve added a number of optional different ways to play Cogmind, many of which even

make significant changes to the experience. You can play them as a challenge, or just to try something different, or perhaps one of these modes is simply more enjoyable for you than the base game. In any case, even if you've already mastered Cogmind, as different experiences these optional features will require that you explore new strategies to tackle new content, mechanics, or even whole new systems. Think of them as official mods, some small, some quite large.

Note that toggling either challenge or special modes will only take effect if starting the game with a fresh new run. In other words, loading a save to continue a run automatically puts the game data in a state for that run, and cannot be changed in the same session, so make sure you're not continuing a previous run if you want to change these options. Or if you discover you had a game in progress and want to switch to your newly toggled mode(s), choose the shutdown option from the game menu to quit and delete the save file, then start up again. This will ensure all game data is properly loaded and compatible with the desired modes.

Challenge Modes

Anyone looking for an extra challenge, or a somewhat different experience, can activate various "challenge modes." These change one or more aspects of a run to increase the difficulty in exchange for extra points. These modes can only be toggled via /user/advanced.cfg, while the game isn't running. Multiple challenge modes can be active simultaneously, except for any incompatibilities stated below. And note that activating some challenges may cause a world seed to produce different results; those that do are marked.

* challengeDevolution: Start with 20 slots chosen randomly (from a weighted distribution), and lose 1, also randomly, at each new evolution. Base integrity never changes, always resetting to 800 with each evolution, and inherent heat dissipation (20) is also static throughout the run. If there are no free slots of the type that devolves, a random excess part is dropped to the ground. Bonus points are applied at each depth change, though the modifiers are negative in the early game before accumulating even larger positive modifiers in the late game. Incompatible with challengeInhibitedEvolution and challengeUnstableEvolution (Devolution takes precedence).

* challengeFragileParts: All parts are destroyed on removal, mimicking the standard behavior of processors. Storage Units are exempt from this effect. Incompatible with challengeStickyParts (Sticky Parts takes precedence).

* challengeGauntlet: All branch access is blocked, as are all main access points except the furthest from your entry location. Waste is still accessible, as are Garrisons, but both always bring you back to the same depth, and only the first Garrison visited at each depth will have working RIF Installers. Incompatible with challengeSuperGauntlet. (Gauntlet takes precedence.)

* challengeInhibitedEvolution: Only evolve half as many slots as usual.

* challengeNoSalvage: Destroyed robots leave no salvageable parts, only matter (and inventory items, for special events that include those).

* challengePureCore: No inventory slots for the entire run. Storage utilities are instead converted to matter when stepped on.

* challengeScavenger: There are no random part stockpiles in the main complex, and any lone items strewn about are damaged. Everything else must be salvaged from other robots, stolen from haulers, or fabricated. Note that static stockpiles are still present, such as those in Storage for example. <affects_seed>

* challengeSimpleHacker: No indirect or manual hacking of any kind.

* challengeStickyParts: No parts can be removed or swapped out manually, and must be destroyed to free their slot(s). Incompatible with challengeFragileParts (Sticky Parts takes precedence).

* challengeSuperGauntlet: All access points are locked, as are all but one garrison access. The only way to advance to the next depth is to find that one garrison and pass through it. Hacking open the entry is guaranteed, and leaving a garrison always takes you upward rather than back to the same map. Waste is still accessible. Note there are no garrisons in -10/-9, and ascension occurs normally there. Be careful around the lone garrison access because it can be disabled from damage, trapping you on a floor and ending the run. RIF Installers are not usable in this challenge.

* challengeTrapped: All non-garrison 0b10-controlled maps that have any traps multiply that number by 10. Also +20% to trigger traps, and trap knowledge cannot be gained via hacks, Zionite intel, 0b10 Decoders, Trap Scanners, or friendly Operators/Minesweepers. <affects_seed>

* challengeUnstableEvolution: You have no control over which slots you evolve; each of your evolutions is randomized, though the chance of evolving a power slot is weighted lower while utility slots are weighted higher.

Special Modes

There have been intermittent special Cogmind events held on specific days, each giving access to alternative content or mechanics, and although these events were automatically triggered on their respective days for players at the time, some of them have been preserved and there is a way to force their activation regardless of the date, making them available even after the fact.

Activating them requires adding a command line parameter when running COGMIND.EXE, "-forceMode:[mode_name]", where [mode_name] is simply the name of the desired mode from the list below. (Within Steam this is handled by going to Cogmind > Properties > Set Launch Options.) If you've done this correctly the message log will show a special announcement when you start a new run. Note you don't actually have to type out the full name of the event, just a subset of it will work, for example simply "-forceMode:Launchers" to get the April Fool's Day event from 2018. If an ongoing event automatically triggers for you due to the current date and you'd prefer to play a run of the regular game, you can do this during the event period by adding the "-noSpecialMode" command line parameter.

Unlike challenge modes, special modes are not necessarily harder, just generally quite a lot different from the normal game, so expect some extreme gameplay changes! Also unlike challenge modes, multiple special modes cannot be combined with one another.

All special modes still available in this version of Cogmind:

* "AFD2018/Launchers" (180401): Every single part on the ground is replaced by a launcher, and bots never drop weapons as salvage, forcing you to find a way to get by in this launcher-filled complex. Oh what will you do?

* "AFD2019/Pay2Buy" (190401): There are no parts to find, and no salvage from robot destruction except matter. Instead your entire source of new parts is the Cogshop, a menu-driven shop available right in the UI and offering instant delivery on payment. To buy parts, earn CogCoins by doing anything that raises the alert level. Prices fluctuate based on demand, and you can also take advantage of timed special deals or even gamble your coins away on loot boxes. You can't use Storage Units in this mode, but your base inventory size is 10 instead of the usual 5. The only thing standing between you and total domination is... money. Go make some at MAIN.C's expense!

* "Halloween2019/Abominations" (191031): A quantum virus is infecting Complex 0b10, and although branches are inaccessible, a new faction joins the fray, bringing with it 20 new robots with a wide variety of new capabilities, as well as a new hidden map complete with boss. Be afraid.

* "Winter2019/RPGLIKE" (191217-200101): Meant as a serious alternative way to play Cogmind with RPG-like XP leveling mechanics and permanent upgrades instead of the standard method of evolution. Gain XP by exploring and/or raising the alert level. You start with much higher core integrity, and 80% of incoming damage to parts (or 40% in the case of Armor-type parts) is instead transferred to your core, thus losing parts is relatively rare. You can also use Protomatter to restore the integrity of your core and parts. Storage Units do not exist and you start with no base inventory, but you can optionally expand inventory capacity via level upgrades. For RIF users, although fewer Relay Couplers will spawn, all that do have double their normal value. Overall quite a different Cogmind experience, but one that might be more enjoyable for those who prefer to have more attachment to their build, or rely on certain rare parts for much of their run without having to worry about protecting or replacing them.

* "AFD2020/Player2" (200401): Take on Complex 0b10 together with a friend! Your new buddy's core integrity is linked to your own, but they also share most of your own capabilities and can attach parts to manage their build and inventory, evolve with each new depth, and have special AI behaviors that make them rather unique compared to other allies. Oh yeah, they can also be quite opinionated.

* "Halloween2020/ForbiddenLore" (201030-201130): It's a Cogmind ARG! This event starts online from forbiddenlore.gridsagegames.com, and has you following hints to various points in the game where you can uncover clues to passwords which can unlock multiple levels of never-before-seen Cogmind lore via the website. Can you reach the highest level? (I can't promise that all the answers will exist permanently beyond the end of the event since being an ARG many exist outside the game itself, but most if not all of it should still be completable for a good while.)

* "AFD2021/Volatile" (210401): Robots explode on death for maximum chaos, and patrols are more frequent lest you run out of things to blow up. AOE weapons are inaccessible, but that's not a problem when your enemies themselves become AOE weapons.

* "Winter2022/Polymind" (221224-230107): Blend into Complex 0b10 as one of their own! Collect Protomatter from destroyed combat bots and use it to take control of almost any other robot. You can't attach or remove parts yourself, or even evolve slots, but by assuming the form and function of other robots you can take advantage of their capabilities to stay under the radar. Find ways to keep your suspicion low, but be wary of especially suspicious enemies. As long as you're not detected, you can waltz right past any 0b10 robots, even swapping places with them. Once detected, either fight it out or try to lose your tail and blend in again by getting back to work as a local. Robots from other factions will always recognize you for what you really are.

Advanced UI

Most of the UI is labeled and self-explanatory, though in some cases extra information is provided in extremely condensed form via abbreviations, unlabeled numbers and letters, or sometimes color.

Tactical HUD

An optional mode that provides more detailed UI info, recommended for all advanced players and roguelike veterans. Activating it in the options provides the following features:

* HUD shows your exact core exposure percentage
* HUD shows both stationary and mobile energy cost per turn, rather than just mobile
* HUD shows full heat breakdown including stationary upkeep, when mobile, and injector/ablative cooling
* HUD movement speed also indicates current momentum
* Movement speed for both self and other bots expressed in time units rather than as a percentage

* Volley speed expressed in time units rather than as a percentage
* Info page for your attached parts with less than 1% coverage shows that value as a fraction instead
* Info page for power sources also includes chain reaction explosion details (loaded on startup)
* Robot part lists show per-item coverage rate
* Robot core damage popups indicate actual remaining integrity rather than a percentage
* Scan window displays both robot integrity and current heat, rather than just integrity
* Multi-wielding melee weapons shows their respective follow-up attack chance(s) in the parts list
* Buttons appear next to part list categories to allow batch toggling (non-keyboard mode only)
* Part integrity markers are boxed upon taking damage, remaining that way until your next action
* Part data visualization shows values next to graph in most modes
* Parts swapped in from the inventory show a line marker to their left as long as the original part still in inventory

All other subsections of this Advanced UI guide assume Tactical HUD mode is active.

HUD

The top-right HUD area shows primary stats and related data.

Energy: The first parenthesized number is energy cost per turn when stationary, followed by total cost per move when mobile. Any energy currently stored in inventory containers is totaled at the end after the "I:" indicator.

Matter: If any attached and active Field Recycling Units are processing matter, parenthesis enclose the amount released per turn followed by the total amount remaining to process. Any matter currently stored in inventory containers is totaled at the end after the "I:" indicator.

Temperature: Like energy, this gives total values under stationary and mobile conditions. If any active coolant injectors are attached, their combined dissipation is shown as a third value.

Movement: The time cost per move is shown in parenthesis, followed by a single digit that represents your current momentum. The base momentum factor usually ranges from 0 to 3, and is increased by repeated movement in the same direction. Higher momentum confers a bonus to melee and ramming damage, thus you'll see your current momentum reflected on the status window damage modifier list.

Scan Window

Hovering over objects displays their basic data here. For robots the top block color reflects their current integrity, the bottom block color indicates their current heat level (not shown when temperature is cool), while the color of the name indicates their current relation with you: red for hostile, gray for neutral, and green for friendly. When spotted by a hostile, a red '!' appears in the bottom left corner. When a robot has some dialogue that requires you to be adjacent or bump into them to hear, a green '?' is shown there instead. The '?' instead appears gray if you've already collected the associated lore and listening to them does not offer any repeat per-run benefits.

Evasion Window

This contains a summary of all defensive modifiers to the chance to be hit by an incoming projectile. It assumes the standard base hit chance of 60%, and cannot take into account the many attacker-sourced and situational factors, thus it is not generally an accurate representation of the actual chance to be hit, instead serving to demonstrate the relative defensive value of certain circumstances and part configurations. Hovering the cursor over the window (or holding '\') opens the full details, which lists each component by name and will also indicate why a given value is grayed out (i.e. inapplicable). Note that while the summary window updates in real time, the details window does not (in the rare case that you're toggling parts while also holding that window open).

Volley Window

While in firing mode, the evasion window is replaced with the volley window, showing summary data for the volley if there are any active weapons. 'R' indicates the range from your current position to the cursor. Letters in brackets refer to all active weapons (using their parts list letter), where those which are not currently within range of the cursor target (and therefore will not fire) are grayed out. For all weapons that will fire, the total resource costs are listed in the second line: time required to fire along with energy ('E'), matter ('M') and heat ('H'). Energy and matter are displayed as total costs, while heat is instead the amount generated per turn over the volley duration.

At the bottom of that window is a button for toggling a visualization of the volley range (alternatively use the 'v' key) to highlight the area reachable by the current volley, using different levels of brightness if multiple ranges are involved. Any visible terrain destroyable using the volley (including generally taking into account all applicable damage modification from active utilities, current momentum, resistances, etc.) will glow red, while any other terrain that could at least theoretically be destroyed by currently inactive weapons or those in inventory will instead glow yellow. The appearance of terrain you don't likely have the means to destroy at the moment remains unchanged. Note that even in such a case, it may sometimes still be possible to destroy the terrain using other means, since the system does not take into account inactive utilities or those in inventory, special weapons, the environment, or other means to creatively destroy terrain, but it is a useful way to quickly confirm that you already do have the capability.

Parts Window

On the top border of this area is current/max mass, and if overweight an "0x#" indicator will appear in which '#' is the number of times you currently exceed your propulsion support limit, a vital factor in movement mechanics. If flying, the overweight indicator appears yellow since you lose your dodge bonus while overweight, and are no longer capable of flying over non-allied bots.

To the right of each part is extra information about each depending on the current mode (coverage/energy/integrity/info). Info ('q') mode is the most esoteric, but provides a useful at-a-glance summary of stats relevant to each type of item, usually enough to make quick comparisons without opening the full info window. The first number for all parts is always the rating, preceded by a * if it's a prototype, or ** if an artifact. A '+' also appears before that to indicate when you currently have a schematic for that type of part, though note that in the case of a few late-game weapons, there may not be enough room in their summary to show the + reminder. Subsequent values differ by item type. All parts except propulsion indicate their mass with 'M'. Power sources indicate energy output with 'E'. Propulsion indicate base cost/move with 'T', and support with 'S'. Utilities show an ABBR/###, where ABBR is a very short four-letter abbreviation for its effect type, followed by the strength/value of its effect. Because each utility has only one primary type of effect, there isn't much point in learning to recognize those abbreviations (though there is a scheme to them). Instead the number is what's really important.

In Vulnerability ('c') mode, the graphs are derived from a combination of coverage and integrity, where those parts with greater vulnerability are more likely to be lost before others, statistically speaking.

In Heat ('e') mode, propulsion values reflect the amount of heat generated by a full turn of movement plus upkeep, and weapon values assume the weapon is fired alone in a 200-time volley with no further adjustments for delay or utility effects.

Before the slot letter for processors and hackware is a dark ':', which is simply an indicator to remind you that removing or replacing that part will destroy it permanently.

When a part is attached via the swap menu, replacing another part, in Tactical HUD mode a line marker appears to its left and will remain there for as long as the part it replaced is still available in inventory. This makes it easier to swap back and forth between a pair of parts, for example if you have a slightly separate

loadout for exploration vs. combat. When the swap menu is opened again for that autopaired part, the previous part to occupy that slot is also listed as part 'Y' for quicker selection. Another feature with much more limited applicability is multiswapping, which allows you to swap all autopaired parts via a single command (Shift-Alt-w). The resource costs are the same, all this does is queue the actions for you. Holding Shift-Alt-a (or hovering the mouse cursor over one of those markers) causes the names of all autopaired parts to appear next to their counterparts, making it easier to quickly determine whether you'd want to do this.

Note that the order parts are listed does not matter, except in the case of weapons, where active weapons in a volley will fire from top to bottom as listed. Reorder weapons as necessary depending on the tactical situation. All parts can be autosorted by subtype with the ':' key, which by default will also sort weapons, but the latter behavior can be manually overridden with the partSortIgnoresWeapons in /user/advanced.cfg.

Inventory Window

The list of inventory items displays used and free inventory space on its top border, and also inherits the current info mode of the parts list.

Sorting by type, accessible via the bottom button/key, can also be reversed by double-clicking/pressing the same button/key.

To tag an inventory item with a text note, use the Special Commands interface (Spacebar) or press Shift-Alt-t followed by its inventory number. This feature is generally used to tag unidentified prototypes and alien artifacts that are known based on other factors or meta knowledge, so that they're not lost in the inventory later. Adding a '!' anywhere in a tag, even by itself, has the additional functionality of forcing that item to require confirmation in order to attach it. Erase an existing item tag by entering an empty tag for it.

Combat Log Window

Among the optional top-center windows is a log containing more specific combat details, the level of which can be set in the options menu. Reading it is not necessary since all vital data is conveyed via the regular log and on the map itself, but some players find additional details useful for learning purposes or exploring optimization strategies. At full detail this will give a complete breakdown of all hit chance modifiers factored into your attacks, represented as a string of abbreviations with the following meanings:

- r range
- s size, target
- sg siege mode
- h heat
- ht heat, target
- u utilities
- ut utilities, target
- m movement
- mt movement, target
- ft flying, target
- im immobile
- ob obstructing robot
- a analysis
- ha hacked targeting
- di disruption (from a hack)
- tc tactical coordination suite

Be aware that internally the game uses decimals to calculate hit chance, so the values shown may be slightly inaccurate due to rounding.

If the log detail is set to High or above, during your turn you can scroll the combat log manually to more

quickly look through the combat history via an expanded view that appears over the map. The expanded combat log automatically disappears if you perform an action, or switch away from the combat log.

If the log detail is set to Full, combat log contents are also mirrored to the left side of the map in order to show more at once, as the action plays out. Use /user/advanced.cfg to adjust the length (mapFullCombatLogMaxLength) and duration (mapFullCombatLogDuration) of the mirrored messages. This feature only applies to the Full UI layout unless mapFullCombatLogOverride is set.

Info Window

When viewing detailed info for a utility, their effect might be followed by one or more of the following indicators:

<stacks>: This effect is combined with those from all active utilities with the same effect.

<no_stack>: Effects from multiple utilities of this type do not stack, only the best available is applied.

<half_stack>: Effects from multiple utilities of this type will stack, but only the best two, where the second best contributes 50% of its effect.

<parallel_ok>: Multiple active utilities with the same long-term effect continuously apply that effect at the same time.

<resume_ok>: If deactivated then later reactivated, this long-term effect will pick up where it left off.

<consumed>: Attaching the utility activates it, using it up permanently.

Map

Various temporary indicators may pop up for robots on the map:

* Numbers that appear next to a robot taking core damage represent its remaining core integrity. This can be reverted to non-Tactical HUD behavior by setting percentCorePopup in /user/advanced.cfg to show percent integrity instead, but in either case the color coding reflects how close they are to core destruction, by percentage. Sometimes more than one number may appear, indicating the same robot's core was hit more than once in rapid succession.

* Combat robots glow red for a moment when they have reached critical heat and are ripe for meltdown by further thermal impacts. The glow only occurs on their turn, however, when they first pass the threshold.

* When first spotted by a hostile robot, a red '!' will flash on them temporarily.

* NPCs with dialogue to speak show a flashing green '?'. The indicator is instead gray, and temporary, if the dialogue is pure lore that you have already read in a previous run. Note that in any of the areas with numerous NPCs to speak to, even among those with green indicators it is possible to distinguish pure lore from other benefits: Those which are roaming around will only provide lore, while lone stationary robots also provide you with something useful.

* Hovering your examine cursor over a known Heavy position for a moment will show their sensor range for reference.

The origins of sound effects heard but not seen are also temporarily marked on the map, and color-coded by type as follows:

Red Combat-related

Orange Door open/close
Yellow Trap trigger, carrier deployment, or garrison dispatch
Blue Emergency door open/close or phase wall destroyed
Green Machine destroyed
Brown Walls destroyed or cave-in

This feature can be disabled via disableVisibleSfx in /user/advanced.cfg.

Open spaces previously occupied by now-destroyed walls and earth have a slight glow to them. Consecutive moves through these unstable areas may cause a cave-in, but only if doing so for extended periods. Tunneling between rooms, or through a wall-earth-wall scenario, will not cave in.

You can export an image of the entire current map via the Special Commands interface (Spacebar), or by using the Shift-Alt-m command. The image is placed in a new /screenshots-maps/ directory and tagged with the map's name, depth, and current map turn number. This process may take a few seconds because it needs to sequentially render numerous frames across the entire map, compiling them together into a single PNG. Images exclude any temporary overlays and animations.

Targeting Behavior

On entering firing mode via the 'f' command, the cursor will automatically shift to the most recent previous target, be it a robot or empty space, and any previous guided waypoints will be recalled (if still in sight). If there is no previous target, or it is no longer valid, the nearest target is chosen instead. Cycle forward and backward through targets sorted in near to far order via the Tab/Shift-Tab, -/=, and Numpad -/+ command pairs. Only armed hostiles are included in the cycle, or if none are visible it will instead cycle through all non-allied targets.

Examine mode uses the same cycling commands, but always includes all non-allied robots, armed and unarmed. When the intent is to attack an unarmed target while other armed hostiles are visible, examine mode can be used as an alternative to cycle through both types of robots, and from there switch directly to firing mode once the desired target is reached.

Note that item examination also allows the same near-to-far cycling scheme, but requires the Ctrl key (Ctrl-Tab/Ctrl-Shift-Tab and Ctrl-Numpad -/+).

Part Auto-Replacement

When attaching parts from either the inventory or directly from the ground, if there are no available slots of the corresponding type, the interface will attempt to automatically pick a part to replace instead. Replacement candidates are prioritized in the following order:

* Replace an item of the same name that has a lower integrity. This process ignores non-functional parts since a player may have other reasons for owning them, so drop those manually if not needed.

* Replace a different part that has the same type of effect, but that effect is not as powerful, regardless of the two parts' respective integrity. This mostly applies to utilities. If more than one part has the same effect type, it will select the one with the worst value, breaking ties by choosing the one with the lowest integrity.

* Replace a different part of the same subtype (ex: "Energy Cannon") where the old part has a lower integrity than any other of the same subtype with a lesser or equal effective rating. "Effective Rating" equals rating, +1 for prototypes. Ties among potential swap targets are broken by choosing the one with the lowest effective rating. This step is skipped by utilities and any parts with an effect, except treads. Weapon subtypes are compared and replaced only if they inflict the same type of damage. And because the only intent behind power source subtypes is to convey a general range of ratings, their subtypes are not treated as separate categories when analyzing potential replacement candidates.

* Armor without any special effect is replaced by other such armor, comparing purely for current integrity (rating is ignored, except to break ties).

If none of the conditions are met, it will simply report "No free slot." The list of rules is relatively short and basic because anything more results in too many complications and lowers predictability. Auto-replacement needs to err on the side of not swapping parts, so it's kept simple and predictable, based on only a few variables. To avoid smart replacement entirely, simply use the manual swap commands, or free the required slot first.

Note that this feature only works for parts that occupy a single slot.

Auto-replaced items are moved to the inventory, or if the inventory is already full, space will either be freed by comparing the item to all those carried based on the same rules above and drop a worse corresponding item, or where no comparison is possible, the replaced item itself is simply dropped.

In addition to attached part replacement, when attempting to pick up an item and put it in a full inventory, the same rules are applied to auto-replace (drop) an applicable inventory item to make room for a better new one. Where no comparison is possible, the pick up action will simply fail with a message indicating the inventory is full.

Item Label Filtering

In addition to the several adjustable itemLabel filters available in advanced.cfg, when mass labeling map items you can also limit the labels to those items belonging to a particular category, namely by each slot type (or non-part items).

After opening labels for all items, use Tab (or -/= or KP+/-) to cycle through the different categories, which will update the labels each time. Labeling categories also obeys the advanced.cfg filter restrictions, so in order to see the entirety of a particular category, if some individual items were filtered out by your options, you'd need to mass label again while the category filter is active.

The category filter will automatically reset to the default (all items) after ten seconds without use, or you can reset it manually by pressing Escape.

Item Search

You can search through all known items on the map via the item search interface, activated via the Special Commands interface (Spacebar), or by Shift-Alt-f ("find"). This opens a new window over the right side of the HUD initially listing all known items, which can be shortened through a wide variety of filters.

Simply typing enters text into the Search bar, where the most basic filter is text which must be found in the item name. For example "Imp" will list all items with the Imp. prefix, but technically also include "Impulse Thruster" (unless a period is added, of course, to search for "Imp."). Combine separate strings (requiring all of them to be present, but not consecutive) by using a comma. For example searching for for "par,char" would return any Particle Chargers. Searches are not case sensitive.

As usual, standard text editing commands are available, such as the Delete key, Arrows to move the cursor, and Ctrl-Backspace to erase the current text. The Enter key also simply resets the current search.

To search based on item properties other than name, special filters are available using the period prefix. For example ".power" will list all items that count as parts equippable to a power slot, or ".treads" will list all tread items. Generally only the first few letters are actually required to specify a particular filter, as indicated in the reference list below, but the full word is fine as well.

Special search filters:

Slots	.po(wer)
	.prop(ulsion)
	.ut(ility)
	.we(apon)
Type	.dat(a)
	.non(part)
	.tra(p)
	.tre(ads)
	.leg(s)
	.whe(els)
	.hov(er)
	.fli(ght)
	.dev(ice)
	.sto(rage)
	.proc(essor)
	.hac(kware)
	.arm(or)
	.ali(en)
	.gun
	.can(non)
	.lau(ncher)
	.spe(cial)
	.mel(ee)
Damage Type	.ki
	.th
	.ex
	.em
	.en
	.ph
	.imp(act)
	.sla(shing)
	.pie(rcing)
	(for launchers, damage type matches the explosion rather than that of the projectile, if applicable)
Integrity	.X%, where X is a minimum percentage of remaining integrity from 1~100
Rating	.X, where X is the minimum rating
Other	.uni(que) retains only items which cannot spawn as random drops

Multiple special filters can be combined with one another, as well as used alongside name filters if desired, again using a comma to delineate separate filters. For example "rifle,.th" would list any items with "rifle" in the name which are also capable of dealing thermal damage. Or ".hov,5,70%" would list all hover propulsion of at least rating 5 with a minimum of 70% current integrity. Notice that it’s not necessary to repeat the ’.’ filter prefix for subsequent filters.

Left-click on a listed item (or use Ctrl-a~z) to center the map view on its location. While the map view is currently centered on a listed item, that item is marked with an asterisk in the list, and the shortest known path to reach it is highlighted on the map. Due to command conflicts not all of the various methods of panning the map are currently available while the search function is active, but mouse users can still hold Shift and move the cursor, and keyboard players can use mapshift mode as normal.

Double-clicking on an item in the list (or repeating its Ctrl command) will automatically move to that item. (You can adjust the detection delay via itemSearchPathfindingDelay in advanced.cfg.) Depending on the

current state of the map and its occupants, however, pathfinding may be interrupted along the way, especially for distant items.

Items are ordered by their current direct distance from your position, near to far, not taking into account walls or available paths. Those items currently within Cogmind's FOV display their distance in blue, whereas those outside FOV display it in a shade of gray depending on distance. To the right of each item is its last known integrity, if available.

Map Comments

Comments can be added directly to any position on the map by activating the dedicated mode via the Special Commands interface (Spacebar) or Shift-Alt-c. Keyboard mode can also directly add/remove comments while in look/targeting mode using c/r. Cycle through existing comments using Tab (+Shift reverses), and edit an existing comment by adding a comment at the same location. Even while outside comment mode, map positions containing comments will be highlighted intermittently as a reminder (this functionality can be adjusted via mapCommentHighlightInterval), and hovering the cursor over an existing comment's position will display that individual comment (if another labelable object currently occupies the comment's position, hovering always shows that by default and toggling all comments would be required to see such a comment).

Lore Collection

In the lore collection interface, lore discovered via terminals is ordered alphabetically by topic, where the category simply refers to its origin. As a shortcut to finding a particular topic, type the first letter of its name to scroll to that section. Below terminal records is dialogue lore, organized by source. Their numbers have no significance and exist simply to serve as a distinguishing identifier for reference purposes. The last group are all robot analyses.

To the bottom right of the lore window is an export feature, convenient for those who wish to read their lore outside the game, share it with others, etc. Left-click a button or hit its corresponding letter key (shift for upper case) to immediately export your current lore to that format. The results are output to your /user/ directory. For export purposes, terminal records are reordered by category, and alphabetized within their category. As with the in-game UI, placeholders are shown for undiscovered lore in the TXT and HTML versions, though the CSV version solely includes your discovered lore. The HTML version is both colored and includes a table of contents with links to each section.

When manually updating to a newer version of Cogmind, records of previously collected lore are only retained across versions if you migrate your old /user/ data, specifically the progress.bin file. (Unless using Steam autoupdates or a -nonportable/custom file path, in which case this is handled automatically.)

Item Gallery

All items attached across all runs are tallied in the gallery, which leaves empty spaces for those which have yet to be found. The parse_system robot hack may also be used on other robots to add their items to the gallery, especially useful when working with allies equipped with unique parts.

To find a particular item more quickly, type the first letter of its name to scroll to that section. Note that this search method ignores name prefixes, e.g. if looking for a "Exp. Transmission Jammer" you should type a "t"; all of the jammers will be next to one another within the "t" section.

To the bottom right of the gallery window is an export feature, convenient for those who wish to review item stats outside the game or theorize about potential builds. Left-click a button or hit its corresponding letter key (shift for upper case) to immediately export your current item stat collection to that format. The results are output to your /user/ directory. Undiscovered items are not included in the export, nor are there placeholders for them. The CSV version is best for number crunching in spreadsheets, the TXT version

lists items with only those stats which apply to them, and the HTML verson is both colored and includes a table of contents with links to each slot type.

When manually updating to a newer version of Cogmind, records of previously attached items are only retained across versions if you migrate your old /user/ data, specifically the progress.bin file. (Unless using Steam autoupdates or a -nonportable/custom file path, in which case this is handled automatically.)

There is currently only one gallery item not available in the base game: If attempting to fill your gallery, you'll need to activate the Winter2019/RPGLIKE special mode and play it to obtain the Protomatter entry.

Achievements

Achievements earned throughout all runs can be browsed, filtered, and sorted in this interface.

All achievement descriptions refer to requirements that must be completed within a single run, except where otherwise noted or obvious. Some descriptions are hidden until earned, in order to avoid plot spoilers, or in a few cases to avoid revealing achievements that are fun to earn at random but otherwise boring to aim for. If a Steam player, it is strongly suggested to avoid third-party achievement managers that reveal all hidden achievements because most are spoilers. There is no need to expose and aim for any hidden achievement, as they are things you'll uncover naturally through regular play.

Achievement pop-up notifications are enabled by default, but can somewhat ruin the atmosphere at various points, especially where plot is concerned, so if you like they can be disabled in advanced.cfg by setting achievementPopupDuration to 0. Similarly, setting showAchievementMessages to 0 also prevents those messages from showing up in the log.

If playing on Steam there is bidirectional syncing of your achievements between each system, so if you reinstall Cogmind elsewhere it'll also download all previous achievements and include them in your local meta data (though in that case you'll have lost some info about your achievements like which game and highest difficulty level they were earned in, because Cogmind stores more achievement-related data than Steam's database). Playing the non-Steam version then later migrating over to Steam will also upload all previously earned achievements that are not yet earned on Steam.

All achievements are equally accessible regardless of difficulty setting. If for some reason you want to remove all your achievements, for example having earned many on lower difficulties and want to try again from scratch on a higher setting, add "resetAchievements=1" to advanced.cfg. This change cannot be reversed so you'd better be sure! The resetAchievements option is only valid for non-Steam players due to automated syncing while connected to Steam.

By default achievements can be earned even in challenge modes and special modes, but some are much easier in these modes compared to the regular game, even unintentionally, so a portion of players prefer to block them except when playing normally. You can do this by toggling "Achievements Anywhere" in the options menu. This setting does not affect achievements that can only be earned during challenge modes.

To the bottom right of the achievements window is an export feature, convenient for those who wish to review their achievements outside the game or maybe process them via other means. Left-click a button or hit its corresponding letter key (shift for upper case) to immediately export your current achievement list to that format. The results are output to your /user/ directory. Note that it only exports the currently listed achievements, so your filters and sorting do affect the results.

When manually updating to a newer version of Cogmind, achievement records are only retained across versions if you migrate your old /user/ data, specifically the progress.bin file. (Unless using Steam autoupdates or a -nonportable/custom file path, in which case this is handled automatically.)

Alt-Tab Oddity

Alt-Tabbing between programs in Windows is known to cause issues with software that requires use of the Alt key. The software doesn't always know the correct state of the Alt key after performing such a command, and therefore may believe the Alt key is down when it really isn't. This can be extra noticeable in Cogmind's keyboard mode, which makes heavy use of the Alt key, by necessity. If you frequently Alt-Tab between programs, on returning to Cogmind you might want to tap the Alt key a couple times to make sure its state has been reset, otherwise you may accidentally enter an unintended command. It doesn't happen every time, but it's worth knowing that this is a possibility which can lead to otherwise confusing results.

Advanced Options

There are a number of special options allowing finer control of certain features, or toggling very specific features some players might find useful. However, these options are only accessible via the /user/advanced.cfg file. As in other config files, binary on/off values are represented as 1/0. Make sure to only edit values while the game is not running.

* exposeKeybinds: Once active, this gives access to the keyboard layout and command rebinding system (both external). Read more about this experimental system in the dedicated thread found in the forums General Discussion board: http://www.gridsagegames.com/forums/index.php?topic=823.0

* blockVi: Blocks vi-key input, to prevent unintended movement from accidental key presses for those who don't use vi-keys anyway. Note that if this option is set, rebinding vi-keys to something else will not be possible!

* showClock: Adds a real-time clock to the top-right HUD area.

* showRunTimer: Adds a real-time run timer to the top-right HUD area, displaying the current run time in hours:minutes. Excludes idle time, which doesn't count towards score sheet's run length tally.

* renderFilter/renderFilterMap: See the later section dedicated to Color Customization under Accessibility.

* challenge???: Use these to activate individual challenge modes. See the manual section on Alternative Rules for a description of each.

* quickStart: When starting a new run, this automatically collects all parts and matter in the Scrapyard and puts you next to the exit. Does not work in the tutorial map or while any challenge modes active. Note if you're aiming for challenge achievements with propulsion-related restrictions, such as Mad Max, quickStart would prevent you from fullfilling those requirements.

* noSelfDestruct: Disables the self-destruct option in the game menu, as well as the corresponding hotkeys.

* noManualSaving: Disables the buttons and commands for accessing the quicksave/load feature in Adventurer/Explorer modes.

* autosaveInterval: Autosaves your progress at most every X minutes, but only once idle for at least three seconds. If 0, will only autosave once at the beginning of each map.

* autosaveCount: Maximum number of previous autosaves to retain for the current run. In the event of a serious error with your save file, you can recover your progress by deleting your current save file (user/save_v[number].sav) and restoring one of the previous states named by order number and time stamp, where the highest number is the most recent. Simply rename it to match the deleted save file

(save_v[number].sav). If 0 will never make backup saves.

* jsonScoresheet: Output local copy of scoresheets in JSON format in addition to the normal text version.

* jsonStatDump: Output local copy of stat dumps in JSON format in addition to the normal text version.

* uploadScoresReminder: If upload scores option currently inactive, force message and audio feedback on startup, and upon entering -1.

* ratingBasedItemSpriteColors: Color items visible on the map by their rating, rather than type. Toggling this off instead uses the ASCII coloring scheme for items.

* itemLabelIntegrityPercentCutoff: When mass labeling map items, only label those with at least this integrity percentage (0 to label always label items regardless of integrity).

* itemLabelRelativeRatingCutoff: When mass labeling map items, skip those without an effective rating of at least the current map level minus this value. "Effective Rating" equals rating, +1 for prototypes. "Map level" assumes -10/Materials is level 1, and increases with each change in depth. For example, a value of 2 will only label items with a rating that at least matches two floors down, or better (e.g. at -5/Factory it will not label items with ratings from below -7/Factory). This setting does not affect non-attachable items or Relay Couplers, which are always labeled. And note that item labels ignore all itemLabel filters if mass labeling within 5 seconds of the previous mass labeling.

* itemLabelIgnoreFaulty: Don’t label any faulty items when mass labeling map items.

* itemLabelIgnoreBroken: Don’t label any broken or malfunctioning items when mass labeling map items.

* itemLabelHideRating: Don’t include item ratings in map item labels.

* itemLabelAddUnidentifiedSize: Map item labels will specify the size of any unidentified items, if greater than 1.

* mapLabelsShowIntegrity: Item and robot popup labels also include the object’s remaining integrity, if not currently at max.

* soundIgnoresDistance: Play all audible sound effects at the same volume, regardless of distance. (Setting has no effect on ambient audio from props, for which volume is always diffusion-based.)

* minimumPropVolume: A value from 0~100, setting the minimum volume at which to play ambient machine audio, regardless of distance. Set to 0 for default falloff behavior. Note that changes due to values greater than 0 are not reflected in the audio log, which still accurately reports volume levels relative to the actual distance, falloff, and other factors.

* examineEntersKeyboardMode: If not already in keyboard mode, entering examine (look) mode via keyboard automatically switches to keyboard mode.

* autoExitKeyboardMode: If examine mode was activated via the examineEntersKeyboardMode option, exiting examine mode will automatically toggle keyboard mode off again.

* noMouseWheelWait: Turns off the mouse wheel scroll-to-wait behavior (can be more appropriate for keyboard players with certain types of laptop trackpads that are easily touched unintentionally).

* autoWait: When attempting to move but lack the energy to do so, automatically wait one turn.

* slowMoveSpeed: Speed (in time units) slower than which movement will be temporarily blocked for 0.5

seconds while a warning message is displayed.

* extremeSlowMoveSpeed: Speed (in time units) slower than which movement will be temporarily blocked for 2.0 seconds while a warning message is displayed. Value should be set greater than slowMoveSpeed.

* movementDelayInterval: Delay (in milliseconds) for each space moved when keyboard running or mouse pathfinding. Minimum is 10ms, and note that the default 100ms is fairly appropriate to be both quick while still allowing time to react to the appearance of something of interest that doesn't normally automatically block movement.

* rulerBlocksMovement: Block movement commands while ruler overlay active.

* invertMapPan: Invert x- and y-axis shift directions when panning map (only affects mouse-based panning).

* disableShiftCursorMapPan: Disable the Shift-Move Cursor map panning feature.

* viewFollowsCogmind: Always centers map view on Cogmind with every move.

* viewCenterpointSensitivity: Delay (in milliseconds) before holding RMB on the map will set a new map view centerpoint. Cannot be set below 100ms.

* viewCenterpointHighlightInterval: Interval (in milliseconds) between highlighting the current map view centerpoint when it is designated at a location other than Cogmind. This has no effect if viewFollowsCogmind is deactivated, or when no alternative centerpoint has been set. Setting to 0 disables this feature. Highlighting is not applied in keyboard mode.

* keyboardMoveViewCenterpointBehaviorFull: Determines how the map view centerpoint shifts while moving in keyboard mode with the map zoomed in. "INSTANT" immediately and fully shifts with each new move direction, "MIDPOINT" is like INSTANT but for 45-degree changes in direction will first shift to the midpoint between the old and new centerpoint, and "GRADUAL" shifts the centerpoint over multiple turns to gradually align with the new direction, also attempting to shift it to include extra spaces inside FOV where doing so won't cause other visible spaces to be moved out of view. "NONE" deactivates this behavior entirely. This setting applies only to the centerpoint behavior under the Full UI layout.

* keyboardMoveViewCenterpointBehaviorModal: As keyboardMoveViewCenterpointBehaviorFull, but applies specifically to Modal/Semi-modal UI layouts, which default to NONE because it is much harder to follow when so zoomed in.

* keyboardMoveViewCenterpointOffsetLimit: Specifies the maximum distance from Cogmind that the map view centerpoint will be shifted while moving in keyboard mode with the map zoomed in.

* centerCursorOnMove: Always centers cursor on Cogmind while moving to properly update the Scan window with the item at the current location (only matters in mouse mode).

* disableMouseMovement: Clicks on the map don't initiate moves or pathfinding in mouse mode.

* shiftViewForEnemies: While the map is zoomed, or while using a Modal/Semi-modal UI layout, if new hostiles to label enter FOV but are currently out of view, the view shifts to include them if possible. "Move Block Duration" must be active in the options menu for this to take effect.

* shiftViewForEvents: If any of certain events occur within FOV but are not currently in view, the view shifts to include them if possible, and pauses input for this duration (in milliseconds). Such events include Watchers sending out distress signals, being spotted by an active Operator, or pinging a special machine.

Set to 0 to disable this feature.

* blockCaveinAreasForPathfinding: Pathfinding never allowed to move through potential cave-in locations unless immediately adjacent to starting position.

* warnOnCaveinMove: Require confirmation when consecutive moves into a new destroyed area might begin to cause cave-ins.

* warnOnMoveOutsideMapView: Moving while positioned outside map view requires confirmation.

* rmbWallsToTarget: Mouse users can right-click on walls to enter targeting mode.

* noMouseWarp: Blocks the cursor from being automatically moved when otherwise convenient. (No effect in keyboard mode, where this happens by default.)

* cursorVisibleDuringAttack: Overrides cursor hiding behavior during Cogmind attacks, keeping it visible.

* cursorAlwaysHiddenWhenInputBlocked: Hide cursor any time input is blocked, such as during any attacks or map event animations. Overrides cursorVisibleDuringAttack setting.

* noRewireRepeatProtection: Disable protection against "over-moving" caused by holding a move key to repeatedly rewire an adjacent robot or trap.

* autoOpenNewIntel: Automatically switch to Intel window in response to gaining new intel.

* warningGlow: Animate UI borders based on current relevant status, where applicable.

* hitChanceVolleyAverage: Hit chance popups appearing next to hostiles in targeting mode report average hit chance for all active weapons in range of target in the current volley, instead of the base hit chance before applying any weapon-specific modifiers.

* percentCorePopup: In Tactical HUD mode, force core integrity popups that appear next to robots taking damage to display as a percentage rather than actual integrity.

* condenseSlotTypeHeaders: In Modal/Semi-modal UI layouts, remove parts list slot type headers when doing so will delay the inventory window from switching to modal form. During this period, categories are instead reflected by bars of alternating brightness along the sides which double as cycle buttons for mouse users.

* partDamageIndicatorsRetained: In Tactical HUD mode, damaged parts retain a box around their integrity indicator until your next action.

* precisePartGraphs: Part list data visualizations also show precise numbers in most modes (Tactical HUD only).

* noPreciseCoverage: Part list coverage visualization doesn’t show precise values, even with Tactical HUD active and regardless of precisePartGraphs setting.

* partSortIgnoresWeapons: Ignore all weapon slots for part list sorting purposes, allowing complete manual control of volley firing order.

* partSortSeparatesDestroyOnRemoval: Part list sorting moves all nonremovable and destroy-on-removal parts such as Processors to the top of their slot category, maintaining their relative order.

* matchSortInventory: Opening any item’s info automatically orders the inventory to list all same-type

items at the top.

* dragDropDelay: Delay (in milliseconds) after left-clicking on an attached part or inventory item before the drag-drop UI kicks in. Note that regardless of delay settings, drag-drop mode is also immediately activated as soon the cursor moves out of the item's button area while LMB depressed.

* disableDragDropAutoEquip: Drag-dropping an inventory part to a non-slot location in parts list will not attempt to find an auto-equip solution anyway.

* showPartRange: Circles the maximum range of weapons (and some utilities) on the map while the cursor hovers over that item in the parts list.

* animateVolleyRange: Animate the volley range visualization when activated (otherwise the indicator appears instantly).

* cursorJumpDistance: Distance in cells for commands such as jump shifting the examine or targeting cursor in keyboard mode.

* forceAsciiMachines: Always display machines using the original CP437 line art, even in sprite mode.

* markUndiscoveredLore: Put a '!' marker before undiscovered lore listed on a terminal.

* hudEnergyPerMoveTurnwise: Replace per-move HUD net energy display with turnwise value including movement.

* hudHeatPerMoveTurnwise: Replace per-move HUD net heat display with turnwise value including movement.

* autoDeactivateSiegeModeOnMove: Attempting to move while in siege mode, or transitioning to it, automatically begins transitioning out of the mode instead of warning.

* autoActivateTreadsOnAttack: Attacking automatically activates any attached treads if possible. If not originally using treads to move, then the next attempt to move will also automatically restore the original movement mode.

* autoReadyLauncher: Activating a launcher automatically deactivates all other non-launcher weapons.

* autoUnreadyLauncher: Detaching or swapping out a launcher automatically activates all non-launcher projectile weapons, excluding datajacks, guided weapons, and autonomous weapons.

* animateMapIntro: Animates the map intro on each new level (i.e. the map scan effect).

* showMapBorders: Reveals all edges of the current map. Although this same information can be learned indirectly by attempting to pan around or use certain other UI features, and therefore it should be on by default, it's a deactivated advanced option because on a few maps using this will result in spoilers. It is suggested to instead consider leaving this off until you have already fully explored the world and are familiar with its secrets.

* highlightCaveinAreas: Fully highlights any visible cave-in areas rather than the low-distraction default, equivalent to holding Ctrl-Alt. Not compatible with LOW_CONTRAST render filters.

* disableTerrainScanAnimation: Players distracted by the flashing of cells revealed or rechecked by an active Terrain Scanner can disable that animation.

* mapCommentHighlightInterval: Interval (in milliseconds) between highlighting the positions of any

manual map comments. Setting to 0 disables this feature.

* modifiedTncPopupDuration: Duration (in milliseconds) for which data windows made available by the Modified TNC will remain visible.

* scrapEnginePopupDuration: Duration (in milliseconds) for which the Scrap Engine resources window remains visible when its contents change, or each time COLLECT mode is toggled on.

* itemSearchPathfindingDelay: The maximum delay (in milliseconds) between commands in order to detect a double-click (or repeated key command) for the purpose of pathfinding to an item search target. Setting to 0 disables this feature.

* activeSensorPathNewDuration: Duration (in milliseconds) over which to fade activity cells newly discovered by an Active Sensor Suite.

* activeSensorPathsDuration: Duration (in milliseconds) for which Active Sensor Suite activity paths remain at maximum brightness. This time is automatically tripled while panning the map.

* alphabetizeCouplerHacks: Sorts coupler-based robot hacks alphabetically rather than by cost.

* messageLogLimit: Maximum number of past messages stored for the message log. Must range from 500~100,000, or set to 0 for unlimited length. Note that high values allow very long runs to create large slow-loading save files.

* combatLogLimit: Maximum number of past message stored for the combat log. Must range from 100~5,000, or set to 0 for unlimited length. Note that high values allow very long runs to create large slow-loading save files.

* combatLogOutput: Output the entire combat log at the end of a run or as part of a manual stat dump. The format is determined by the message log output setting in the options menu, so requires that to be active as well.

* noFluffLogMessages: Don’t show a handful of potentially repetitive log messages meant mostly for atmospheric purposes.

* machineAnalysisLogsNoninteractive: Machine Analyzers will also report non-interactive machine discoveries to the message log.

* disableMessageLogTurnNumbers/disableCombatLogTurnNumbers: Don’t prefix log lines with the number of the turn on which that event occurred.

* reportBotnetSupportedSuccess: Successful hacks that would’ve failed without current botnet support are indicated after hacking.

* reportBotnetInsufficientFailure: Failed hacks that would’ve succeeded with more botnet support (up to a maximum of three linked Terminals) are indicated after hacking.

* showBuildType: Cogmind will intermittently analyze your current build and assign a name to it, displaying it above the parts list. Updating is lagged to make it more reliable.

* showAchievementMessages: Message log achievement messages can be turned off to retain the original more immersive experience.

* achievementPopupDuration: Duration (in milliseconds) for which each achievement popup will remain visible over the map. Setting to 0 prevents the popups from showing at all (this also affects whether any

show on the game over screen).

* mapFullCombatLogMaxLength: Determines the maximum number of combat log lines automatic displayed at once on the left side of map (requires full combat log detail setting in options). The value must be between 10 and 48 (inclusive), though setting to 0 deactivates on-map log mirroring completely. Only compatible with Full UI layout unless mapFullCombatLogOverride set.

* mapFullCombatLogDuration: Duration (in milliseconds) before a given line of the above on-map combat log data will disappear.

* mapFullCombatLogOverride: Always display automatic combat log lines on the left side of map regardless of combat log detail level or UI layout.

* mapMessageLogMaxLength: Determines the maximum number of message log lines displayed at once at the top-left of the map while using the Modal UI layout. The value must be between 6 and 10 (inclusive).

* mapMessageLogDuration: Duration (in milliseconds) before a given line of on-map message log data will disappear.

* mapCombatLogMaxLength: Determines the maximum number of combat log lines displayed at once at the top-center of the map while using the Modal UI layout. The value must be between 6 and 10 (inclusive). These messages are not shown in that space if mapFullCombatLogOverride active.

* mapCombatLogDuration: Duration (in milliseconds) before a given line of on-map combat log data will disappear.

* mapAlertDuration: Duration (in milliseconds) for which a given ALERT log message mirrored to the map will remain visible.

* pauseWaitDurationForEnemies: Number of milliseconds for which to block all wait commands after seeing a new enemy (or only non-disarmed threats if "Stop on Threats Only" option is on). Enemies that move in and out of view will not trigger the block until they haven’t been seen for at least 10 turns.

* pauseMoveDurationActiveSensor: Number of milliseconds for which to block movement on coming into range of an enemy’s Active Sensor Suite, or 0 to deactivate this feature.

* activeSensorWarningBufferRange: Distance outside normal Active Sensor Suite detection range beyond which to reset the warning animation and movement block for a given hostile active sensor user, allowing it to take effect on approaching that enemy again.

* flashProjectileVictims: Robots hit by projectiles flash briefly.

* cogmindPartLossMapIndicator: Choose a style for how to represent Cogmind’s own part loss on the map. Defaults to NONE, but other valid options are "COLOR" (flashes red), "X" (flashes a red X), and "NAME" (displays the name of the lost part).

* alwaysWarnAboutResearchers: Warn about Researchers that will witness an attack even if you’ve already been scanned.

* remindCorruptedInventoryOnExit: Warn when about to exit the current map and a corruption-clearing evolution will/might occur but inventory currently contains unattached corrupted parts.

* ignoreAscendConfirmation: No confirmation required to exit the current map under normal conditions, but does not block other types of warnings like leaving allies behind.

* ignoreNeutralMeleeConfirmation: No confirmation required for normal movement-based melee attacks against neutral or inactive bots (no affect on force-melee behavior).

* disableSecondPropulsionAutoactivate: Attaching a second (or third, etc.) form of propulsion when another is already active will not autoactive the new form.

* disableVisibleSfx: Deactivate visible sound effect animations outside FOV.

* disableInfoWindowCloseAnimation: Close robot/item info windows instantly instead of animating their fade.

* disablePathfindingRobotAvoidance: Mouse pathfinding movement normally avoids blocking non-hostile robots, but disabling it instead stops when blocking, allowing manual redirection rather than letting the game choose the fastest route.

* disableManualHackingHelp: Deactivate hacking autocompletion and all automated manual hacking codes assistance.

* disableAutoBackups: Do not make daily backups of the /user/ directory contents.

* disableEscMenuAccessMouse: In mouse mode, prevents the Escape key from opening the game menu (F1).

* disableEscMenuAccessKeyboard: In keyboard mode, prevents the Escape key from opening the game menu (F1).

* disableSteamRichPresence: By default, if you’re playing via Steam then friends will see your current build type, location, and status in game. Setting this to 1 will disable that feature.

* muteAudioOnFocusLoss: Automatically mute all game audio whenever app focus is lost, for example due to Alt-Tabbing or clicking on a window other than Cogmind. Audio will be restored on returning to the game.

* muteTextSfx: Silence the standard text/printing sound effects.

* muteGlobalAlertSfx: Silence alert sound effects played for any non-plot-related global alerts with on-map indicators, such as extermination dispatches, garrison responses, and assaults.

* audioLogIncludesFovSounds: Include all non-ambient sound effects even when the origin is currently in view. Note that even when included, the audio log still excludes non-ambient sounds emitted from Cogmind’s position, as these are generally Cogmind’s own actions and attacks.

* audioLogIncludesFluffMachines: Include even ambient machine sound effects for those that are mostly fluff, as in they are non-explosive and serve no other special function.

* audioLogMaxLength: Determines the maximum number of sound effects listed at once, where if the list is full when a new sound needs to be added, the oldest non-ambient sound will be removed to make room. The value must be between 5 and 48 (inclusive).

* audioLogDuration: Duration (in milliseconds) before a given non-ambient sound effect entry will disappear from the audio log.

* resetAchievements: This value is not present in the file, but may be added manually and set to 1 to erase all previous achievement records. This effect is permanent and you’ll have to start from scratch

again! Only applies to non-Steam players, since the Steam version automatically syncs achievements again once logged in.

System Options

system.cfg also contains a few settings which can be used to manually adjust system-related features:

* mapWidth/mapHeight: These change the dimensions of the map view, in spaces, which in turn affects the size of the entire UI and game window. While the initial values are automatically calculated to give you the maximum font size for your current display, you technically have the option increase the dimensions and see more of the map at once, reducing your maximum available font size as a result. Changing the mapWidth is also a way to convert the game to a 4:3 window on a 16:9 display, in order to have other apps visible beside Cogmind. (Cogmind was designed to work with 4:3, so there are no compromises necessary for that change.)

* fpsCap: Cap your FPS to the specified value. 0 can be used to completely uncap it.

* showFps: Displays current FPS in program title and on in-game HUD (can toggle via Ctrl-F3).

* noAudio: Prevents loading of any sound effects on startup. This may reduce startup time, though also means sound effects cannot play regardless of mute or volume settings. It also prevents the audio log from being displayed. With this setting enabled, if you don't intend to ever use audio you can also safely remove or delete Cogmind's entire data/audio/ directory and the game will load and play normally, just without sound, and total file size is reduced by 35 MB, bringing it to a mere 13 MB.

* unlimitedFontSize: Experimental feature for special monitor setups only! Cogmind can only start up on a system's primary monitor, thus in cases where a secondary monitor has a resolution suitable for a larger font than the primary one, the game cannot take advantage of that in cases where players want to use their secondary monitor. If you have such a setup, set this value to 1 and you will have access to all font sizes regardless of resolution. Then you can set a larger font and once Cogmind is open on the primary monitor, use Shift-WinKey-Left/Right to swap it to the secondary one. If your monitors do not share the same aspect ratio, you may also need to adjust mapWidth/mapHeight accordingly to fit the game on the screen. Unfortunately the library on which Cogmind is built cannot support "fullscreen mode" using a resolution larger than the primary monitor, so this feature is only supported in windowed mode. While in fullscreen mode, you will be unable to select oversized fonts from the options menu. You can set the font size and name in the system.cfg fontSet entry, or within the game while in windowed mode, though because your window may also become too large to access that menu via mouse, know that Ctrl-PgUp/Dn can also be used to cycle through all usable fonts. Overall not ideal but it might satisfy some player's needs so I've exposed this former debug setting as a feature.

File Options

Normally all files are saved within the Cogmind directory, and you can even play it off a USB stick, but some players may want the option to store their saves and other files elsewhere.

You can specify a save path by adding "-customFilePath:PATH" to the COGMIND.exe command line parameters, where PATH is the absolute or relative file path under which to store all settings, scoresheets, screenshots, and the backup archives. Alternatively, the "-nonportable" parameter can be used to automatically assign the path based on the Windows default: "C:\Users\[USER]\.Grid Sage Games\Cogmind"

Note that custom file paths are not compatible with Steam Cloud, and like other game-related paths cannot include non-ASCII characters.

Accessibility

One of Cogmind's original system requirements was based on the fact that its content is designed around ideally having a large number of grid cells visible on the screen at once, thus it was built with the expectation that the device it's played on has a very large screen. As a terminal interface, Cogmind scales directly with physical screen size (resolution is irrelevant), so the best Cogmind experience is had with a suitably large monitor.

That said, numerous accessibility features have been added over the years, both to aid players who may not have access to such a device, and also those with various other needs. Read about these below.

UI Layout

There are three different UI layout settings available in the options menu, offering a way to adjust the visual balance between text/tile size and the amount of information displayed at once. Changing this setting requires restarting the game to take effect.

* Non-modal: Cogmind's original terminal interface layout was designed to target an information-dense 60 rows, providing the largest map view area (>50x50) and access to all main windows at once without having to open them separately. The resulting text size is also therefore the smallest, suitable for large monitors, although map zooming can be most useful in this layout on some devices.

* Semi-modal: Similar to the non-modal 60-row layout, but with 33% larger text/tiles and a smaller map view (usually 50x35). Also once you have evolved many part slots the inventory window will become modal, meaning it must be opened separately to interact with its contents, although smart autotoggling features take effect under the right conditions. This middle-ground setting is the default.

* Modal: This more extreme setting similarly allows for a larger text/tile size, but in order to further expand the map view (closer to 50x50) all top-side consoles are hidden by default (i.e. the message log, allies menu, intel menu, and combat log). Log contents are temporarily mirrored to the map view, and the related menus and windows can be opened separately if and when needed. Advanced.cfg options allow you to adjust the number and duration of messages shown.

Map Zooming

The size of the tiles and most text in the map view area can be doubled by toggling with the 'z' key or ZOOM button at the top of the map, or by using the mouse wheel while panning the map with Shift. Considering the scale of Cogmind's maps and ranges at which visual detection, sensor operation, and ranged combat occur (the ranges around which it was designed), playing while frequently zoomed in can be more challenging overall, for which several QoL features were added to help mitigate some drawbacks.

One of the main challenges of having a smaller zoomed view area is the inability to immediately see new terrain and objects coming into view in your direction of exploration. To focus more on a particular direction, hold RMB with the cursor over a map position to designate that as a new relative centerpoint, and all subsequent movement will maintain focus at that same relative direction and distance. Modify this point as often as you like given your needs, or hold RMB on Cogmind to restore the normal centered behavior.

Non-mouse players cannot as easily select distant cells, so by default whenever zoomed in keyboard mode, regular movement will automatically attempt to adjust the map view so that a more useful or pertinent area is visible at all times. Aside from the default behavior, there are also several other types of automation to choose from. Read about the keyboardMoveViewCenterpointBehaviorFull advanced.cfg

option for how to switch to another, or deactivate this feature entirely. None of these methods will be perfect for all situations, but they might save time. They can also feel quite strange at first, but with some experience they might be something you can get used to, or perhaps figure out alternative approaches that work for you using manual assignment.

As with the mouse, keyboard users can also manually designate a relative centerpoint. This is done with the 's' key in examine mode with the cursor over your desired target cell. Assigning a manual centerpoint deactivates the automation feature until designating Cogmind's own position to reset it, at which point the automated system will again take over. This feature is also useful for quickly recentering the map to reset the automated system, by simply pressing 'x' to enter examine mode followed by an immediate 's'.

Since you can only see so much at one time while zoomed, only a quarter of the normal map area, there is a good chance that relevant nearby objects will be somewhat beyond the the edges, including even objects otherwise within Cogmind's own FOV. While manually panning the map or designating a centerpoint to keep an eye on a particular direction is useful, and temporarily zooming out to get a general idea of the surroundings is another option, to save time there are a variety of new markers that can appear at the edges of the map view to denote objects in that direction. You'll see markers for offscreen hostiles, non-hostiles (both neutral and friendly), new items, and sensor data. Each marker category uses its own unique background color, and in the case of sensor data is also darker and blinks in and out to reflect the fact that it is not a directly visible object like the others. Items are only marked until automatically labeled for the first time after entering view.

To aid in quickly drawing attention to new incoming threats, if they are visible to Cogmind but not in view at the time the map will automatically shift to show them and pause input for a moment. This behavior can be disabled via the shiftViewForEnemies advanced.cfg option. The map will also shift for certain other events you likely want to be aware of, including Watchers sending out distress signals, Operators preparing to report you, and special proximity pings of nearby machines (all controlled by a separate option: shiftViewForEvents).

Although available even while not zoomed, the <HOSTILES> button that appears above the map while there are hostiles within Cogmind's FOV was added primarily to facilitate play while zoomed in, offering another way to know and confirm that there are hostiles in line of sight, even if not currently in view. The button reports the total enemy count, and can be pressed to focus on and highlight the largest concentration of enemies. Pressing it again restores the centerpoint to its original area, or in some cases if there are other enemies not visible in the first highlight area it may be able to determine that and focus there instead.

Font Styles

A large range of manually adjusted font styles have been provided in addition to the default font. Cogmind's default font may be less readable for some players, a side effect of all stylistic pixel fonts, so you may consider switching to another one in the options menu. Terminus is an example of a font specifically designed for readability at smaller sizes, so if necessary you can try that option.

Cogmind already starts up using the largest font size available for your current screen (as far as whatever relevant information the device provides it on startup, anyway), so in adjusting this you'll be looking to check out other fonts of the same size but different style. They should be at the top of your list, but the entire list including even smaller sizes is still provided because some people like to play in a smaller window or with a smaller font size combined with an even larger map view.

Tiles vs ASCII

In terms of the map view, while the tiles have been designed with ASCII-like sensibilities, some players have found that in their situation on only borderline acceptably large screens, switching to ASCII mode is a good way to improve readability. Although the majority of players do use the default tileset, Cogmind

was designed to be played in ASCII and facilitates this process with useful interface features like an automated labeling system for map objects.

Currently all font sets available by default in the options menu share the same map ASCII style, but for extra readability in ASCII mode some map fonts can be converted from their default appearance to another style through manual activation. Available font sets are controlled via data/fonts/_config.xt, so to add any of the following alternatives, open that file and activate any additional fonts you'd like to try by removing the two slashes at the beginning of their corresponding line to allow them to appear on the in-game list:

"T_" is Terminus, a thin and readable sans serif font
"VGA_" is IBM VGA, one of the classic DOS fonts
"V_" is Verite, a fat mostly-serifed font
"K_" is Kaypro, a thin-stroked stylized techno font

Although these settings only affect those who use the indicated font sizes (fullscreen or windowed), more map ASCII alternatives can be added upon request if compatible with the desired size.

Difficulty Modes

Although Cogmind was designed and balanced for its hardest mode, naturally some players simply don't have the time or inclination to strive for mastery, thus alternative modes are available that tweak multiple aspects of the game to make survival easier. You can read much more about these modes and their full effects under the Difficulty section towards the beginning of this manual.

Some players prefer to explore the world of Cogmind using more typical RPG-like XP leveling mechanics and permanent upgrades instead of Cogmind's standard evolution and attrition systems. Originally created as a special event in 2019, "RPGLIKE" is still available and advertised as a serious alternative for those who prefer that sort of gameplay. This special mode significantly changes the experience, and in most respects is generally much easier, but has at least allowed some players to get more of what they want out of their time with Cogmind. You can read more about RPGLIKE and its implications, and how to activate it, under the Alternative Rules section of this manual.

Custom Cursors

By default Cogmind uses a 16px hardware cursor, but that may not be suitable at extremely high resolutions depending on OS cursor scaling settings. If the cursor appears too small (or for some other reason you'd prefer a different style), Cogmind includes a way to display a much more flexible software cursor.

Simply put a file named "cursor.png" in the /rex/ directory and it will be detected automatically on startup. Any image dimensions are accepted, and any pixels that use the hot pink color (255,0,255) will be transparent (see samples provided in that directory). The top-left corner (position 0,0) is always considered the "hotspot," or where the click is actually detected.

Note that a software cursor may feel slightly sluggish compared to a hardware cursor if FPS is low, or even while maintaining the normal 60 FPS cap. Using the system.cfg option to raise the cap will likely resolve the issue as long as your CPU is fast enough to push the frame rate even higher.

Colorblind Mode

Toggling Colorblind Adjustment in the options menu makes various color-based adjustments to parts of the interface that may help certain players.

* Neutral 0b10 bots show as light gray instead of green
* Most orange UI colors instead appear azure blue
* Most green UI colors are converted to light gray

* HUD Core/Energy/Matter readout uses higher contrast colors

While some effects will take place immediately on toggling this option, fully applying it requires a manual restart.

Consider combining Colorblind Adjustments with the BRIGHTNESS renderFilter described below, or try SHIFT_HUE.

Color Customization

Custom UI and map color schemes are currently controlled via two advanced.cfg settings: renderFilter and renderFilterMap. These can be used to specify one or more filters that adjust the appearance of the map and/or UI as a whole. For example, renderFilter=BRIGHTNESS(80) will set the brightness of the entire game to 80% of what it is by default. Or renderFilterMap=BRIGHTNESS(80) will only darken the map.

Each setting supports mixing as many filters as you want, though remember that each additional filter requires more processing time per frame. Combine filters by separating them with '|'. E.g., renderFilter=BRIGHTNESS(90)|SATURATION(90) lowers both the brightness and saturation by 10%. If there are multiple filters they are applied in the order listed, and all map filters are applied before any program-wide filters.

The available filters are as follows:

* NONE: The default appearance.

* BRIGHTNESS(X): Set the color brightness to X, usually a value from 1~100. Perceived color brightness is not linear, but adding in more complex calculations to compensate would slow it down so this is a direct percentage modifier. As a reference, setting to something like 70-80 is darker but maybe not too dark, depending on your monitor. Values above 100 will increase brightness, for example 150 makes everything 50% brighter, 200 doubles brightness, etc.

* SATURATION(X): Set the color saturation to X, usually a value from 0~100. Most colors in Cogmind are fully saturated, and this value can be used to take the edge off. As a reference, try something like 66.

* SWAP: Swaps the foreground and background colors. While the UI generally looks terrible in this state, it may be your thing when applied to the map, or at least fun to toy around with.

* SHIFT_HUE(X): Shift/rotate the hue of the entire palette by X, a value from 1~360. Blacks stay black, grays stay gray, but all other colors will become something else. For example, a value of 180 will create a mostly pink UI, 90 is blue, and 270 is orange. Negative values can be used, and will automatically be converted to the proper value within the range.

* SHIFT_RGB(X): Shifts all RGB color values by X, a number from 1~255. The shifting will ignore any blacks, but this filter is terrible for pretty much anything except creating... color chaos. Negative values can be used, and will automatically be converted to the proper value within the range.

* INVERT: Also known as the negative effect, this inverts all foreground and background RGB values. This mode results in a light-themed interface, and can look decent when combined with other modes such as low-contrast (resulting in a brighter pastel effect) and possibly map swapping. However, inversion works much better in ASCII mode instead of with tiles due to the shading method used in the tiles art.

The following additional filters are only applicable to renderFilter, not the map-only version:

* AUTUMN: A custom low-contrast mode that converts black backgrounds to dark maroon, and raises the

brightness of all darker foreground colors while also slightly darkening the brightest foreground colors to take their edge off. In this mode some of the ASCII item art may look a little odd (and not as good), as will parts of the intro and ending animations, but this only affects the minority of areas that use their own background colors. Further time might be invested in overcoming some of the remaining aesthetic challenges of implementing low-contrast modes if it seems like they're being used.

* SLEEPY: A custom low-contrast mode like AUTUMN but with a comfortable blueish background.

* LOW_CONTRAST(R,G,B): Aside from the AUTUMN and SLEEPY presets, you can also use this filter to create your own low-contrast mode by specifying the RGB values of the desired background color. It should be something relatively dark.

Note that any filter changes affecting color that are made while a run is in progress will in some cases cause areas outside FOV to appear strangely until leaving for a new map.

Audio Log

Off by default but available in the Audio section of the options menu, activating this log lists sound effects at the top-right corner of the map view. Ambient sounds are always listed, while non-ambient sounds are only shown when the origin is outside Cogmind's FOV. Ambient sounds are listed first, followed by a separator, then any non-ambient sound effects.

The audio log is meant to be an accessibility feature akin to closed captions, allowing anyone who keeps their volume low or muted to be able to retain access to important audio knowledge (alongside the visible SFX feature). A given sound's volume must be at least 10% to be included in the log, since anything lower is not generally audible anyway, and it keeps the log cleaner and more focused.

In choosing what sounds to report in the audio log, in most cases it was geared towards information that is strategically helpful to know when out of view, rather than providing an exhaustive list. That said, almost no door actions are reflected, since even if out of view they are clearly displayed via the visible SFX system, and their frequency is often rather high. An exception was made for phase walls, since those are strategically important but do not reveal their precise location as a visible SFX source. Other sounds that have clear visual alternatives are also generally excluded to avoid adding unnecessary "noise" in the audio log, for example dispatch alerts, and the EMP charging and discharging in Waste (which come with their own graphics and messages).

Note that weapon sounds do not necessarily list the specific weapon name, but instead the base name for the sound. For example there are a number of railgun variants that use the same sound, and the audio log will not differentiate between them, simply listing each as a "Railgun," but any weapons with unique sounds are indicated by their specific name where applicable.

There are several advanced.cfg options for customizing the audio log behavior. Aside from those, be aware that using the noAudio system.cfg setting, or muting the audio (for example via Ctrl-F11), will also automatically prevent the audio log from displaying any sounds. You can, however, simultaneously achieve a muting effect and retain audio log information by setting the Master Volume in the options to 0.

Advanced Options

In addition to the normal options found in the in-game options menu, over 100 more specific options have been implemented to allow further tweaking of the interface appearance and its behavior. You can read more about these and how to adjust them in Advanced Options under the Advanced UI section of this manual.

Advanced Commands

While it's recommended to use the in-game basic and advanced command lists that show commands within the window to which they apply, commands are also listed here as an alternative way to reference them (and so that they can all be viewed externally in the text version of this manual).

General

Esc \ RMB	Close/Cancel
0	Clear/Previous message
Esc \ ? \ F1	Help/Options
Ctrl-Shift-Alt-s	Save & quit
Ctrl-+/-	Global Volume Up/Down
Spacebar + s \ Shift-Alt-s	Dump current stats
Alt-F10	Self-destruct/Restart
F2	Toggle keyboard mode
F3	Toggle ASCII mode
Ctrl-F3	Toggle FPS display
Ctrl-F4	Toggle cursor forced centering
Ctrl-F5	Toggle auto-waiting
Ctrl-F8	Create save point (Adventurer/Explorer)
Ctrl-F9	Load save point (Adventurer/Explorer)
Ctrl-F10	Toggle part auto-activation
Ctrl-F11	Toggle audio
PrtScn \ F12	Screenshot
Ctrl-F12	Toggle mouse warping
Alt-Enter	Toggle fullscreen

Map

Alt-Move Keys \ Accent + Move Keys	Shift map view (vi keys require +Shift outside mapshift mode)
Enter \ Alt-KP5 \ CMB	Center on self (repeat cycles through drones)
Shift-Move Cursor	Pan map
RMB (Hold)	Set view centerpoint
Accent	Toggle mapshift mode (movement keys shift map)
v \ Ctrl-CMB	Show volley range
Spacebar + x \ Shift-Alt-x	Toggle ruler overlay
Shift-Alt-o (Hold)	Show orders
1	Label hostiles
2	Label friendlies
3	Label items
Tab \ -/= \ KP-/+	Cycle item label category (+Shift reverses tab)
4	Label exits
Ctrl-Alt (Hold)	Highlight path to cursor, or set keyboard path

and any cave-in areas
x Examine (look) mode
Spacebar + f \ Shift-Alt-f Find item (search)
Spacebar + c \ Shift-Alt-c Map comments
Spacebar + g \ Shift-Alt-g Go to examine cursor
(outside examine mode
repeats last destination)
z \ Shift-Wheel Zoom Map
m Map intel overlays
Spacebar + m \ Shift-Alt-m Export known map as a PNG
Backspace \ F9 Show world map

Movement

KP# \ LMB \ Arrows \ vi-keys Move
(Arrows: +Shift/Ctrl
for Diagonal)
Shift-KP# \ r-Arrows \ r-vi Run (r/Esc cancels)
< \ > \ LMB Up stairs / To branch /
Trigger/Install trap /
Release drones

Control

KP5 \ . \ Wheel Wait
g \ LMB Get item
a \ Ctrl-LMB Get item and attach
f \ LMB Fire (+Ctrl to force)
Ctrl-Shift + Movement Keys Force melee attack
Ctrl-Shift-LMB Force melee attack
s \ RMB Status
o \ Shift-RMB Order ally
o \ KP5 \ Enter \ LMB Confirm order target

Look/Target

Movement Keys Move cursor (KB mode only)
Tab \ -/= \ KP-/+ Cycle targets
(+Shift reverses tab)
Ctrl-Tab \ Ctrl-KP-/+ Cycle items
(+Shift reverses tab)
d \ KP0 \ RMB \ Ctrl-RMB Data (info)
Backspace \ KP Enter Center cursor on self
s Set view centerpoint
(KB mode only)
c/r Add/Remove map comment
KP5 \ Enter \ LMB Set guided weapon waypoint

HUD

\ Evasion breakdown (hold)
Spacebar + i \ Shift-Alt-i RIF abilities
Ctrl-F7 Toggle tactical HUD

Parts

Shift-a~z \ RMB Info
Ctrl-a~z \ LMB Toggle state
Ctrl-Shift-a~z/1~0 Swap/Reassign
Drag to Slot Swap/Reassign
/ + a~z \ Ctrl-RMB Inventory swap/attach
Spacebar + r \ Shift-Alt-r Reswap
Alt-a~z \ Ctrl-LMB Detach
Drag to Inventory Detach
Drag to Map Drop
, + Alt-a~z Detach-drop
d + a~z/1~0 Modal remove/drop (KB mode)
Spacebar + p \ Shift-Alt-p Purge all parts
; Cycle propulsion mode
" Toggle utilities w/upkeep
' Toggle all weapons
: Sort by subtype
p Modal part management
Shift-Alt-a Show all autopairs (hold)
Shift-Alt-w Multiswap all autopairs
c Visualization: Coverage
(toggle for vulnerability)
e Visualization: Energy
(toggle for heat)
w Visualization: Integrity
(toggle for mass)
q Visualization: Info
(toggle for matter)

Inventory

---

i Open/close inventory
(for modal layouts)
]/[\ Wheel Scroll (+Ctrl for page)
t Type Sort (again to reverse)
Ctrl-1~0 \ LMB Attach
Drag to Parts List Attach
Shift-1~0 \ RMB Info
Alt-1~0 \ Drag to Map Drop
/ + 1~0 \ Ctrl-RMB Swap
Spacebar + t \ Shift-Alt-t + 1~0 Tag item with a custom name
Ctrl-F6 Toggle auto-sorting

Message Log

KP-/+ \ Wheel Scroll
PgUp/Dn Page up/down
End End
F4 \ LMB/RMB Expand/Shrink
Spacebar + z \ Shift-Alt-z Add Note to Log

Extended Log (F5)

-/= \ Wheel Scroll

Allies (F6)

o	Order mode
-/= \ Wheel	Scroll
o + Shift-0~5 \ RMB	Info
o + 0~5 \ LMB	Order
r	Order all repairs
o \ Esc	Cancel order mode
Shift-Alt-o (Hold) \ Hover [ALL]	Show orders

Intel (F7)

m	Map intel mode toggle
-/= \ Wheel	Scroll
Shift-0~5 \ RMB	Info
0~5 \ LMB	Toggle intel
m \ Esc	Cancel map intel mode

Combat Log (F8)

-/= \ Wheel	Scroll

Text Input

Characters	Enter text
Left/Right/Home/End	Move caret
Delete	Delete character at caret
Backspace	Delete last character
Ctrl-Backspace	Clear text
Up/Down	Cycle text (if applicable)
Tab	Autocomplete (if applicable)
Ctrl-v	Paste from system clipboard
Enter	Confirm
Esc	Cancel

Keyboard Logic

While all features are accessible via mouse, using the keyboard can make for much more efficient play. Thus it is helpful to at least gradually learn some or all keyboard commands. Fortunately there is a general logic to the command modifiers that makes them easier to remember and keeps the input scheme consistent throughout the game:

- Ctrl: (Parts) Attachment/activation ("up")
- Alt: (Parts) Detach/Drop ("down")
- Shift: Information
- Shift-Alt: Special commands, most also accessible via Spacebar
- F-keys: Interface mode changes
- Escape: Cancels/exits pretty much anything

Anyone preferring to manage parts via keyboard input without using any Ctrl/Shift/Alt modifiers can access all the same functionality via the modal part management menu. Press ’p’ to activate the menu and see the full list of options, then select a submenu option and appropriate target letter or number. In this mode Drop always refers to a direct drop to the ground, even if there is sufficient inventory space to hold a dropped attached item. Use Remove to instead attempt to put it in inventory first, or on the ground if not enough capacity.

Key Holding

Certain instances in which it might be preferable to hold a key to repeatedly perform an action can be safely done until the desired effect is achieved, without negative consequences. Examples of possible uses for this behavior:

- Holding a movement key to rewire a robot or trap (1-second block)
- Holding wait while a visible fabricator or repair station completes its task (2-second block)
- Holding wait while waiting for a hostile to enter view
- Holding wait while waiting for a System Backup Module to reduce system corruption to 0%
- Holding wait while waiting for a Recalibrator to fix a broken part
- Holding wait while using Protomatter in the RPGLIKE special mode

---

Notes

Did you know Cogmind’s manual contents appear both in game and as a text file "manual.txt"? You can read the manual outside the game, or even edit the file and have the changes appear in game as well. However, for convenience and because automated updating via Steam would overwrite your modified manual, in order to facilitate maintaining your notes across versions you should instead write them in the provided user/notes.txt file. (Only basic ASCII characters are accepted; foreign language characters are ignored because the engine cannot display them.)