PROJECT: Dead Channel
VERSION: v.01
LAST UPDATED: 1/27/2026

TABLE OF CONTENTS

1. Vision

You're a mercenary hacker in a gritty cyberpunk world. Your job is to assemble a team and support them remotely while they complete their objectives. You're the invisible hand that makes impossible heists possible… but only barely.

Elevator Pitch: 2 parts "plate spinning" time management + 1 part roster management + a dash of autobattler

Inspirations: Shadowrun (particularly the Genesis game), Darkest Dungeon, Rhythm Doctor (hear me out)

2. Core Loop

The game has 2 primary layers: the Run layer (individual missions, core moment-to-moment gameplay) and a Management layer (building and advancing your team and your personal tech). Primary focus at this point is the Run layer.

Run Layer: The screen is divided in two parts. On the top is a horizontal track that abstractly represents the runner team and their path toward their objective(s). The runner team is represented on the left, and obstacles and other interactables will scroll toward them from the right. The player can interact with objects on the track to hack them, which usually will involve one of several kinds of short puzzles, which will appear as moveable popup windows. Once hacked, obstacles can be temporarily disabled or altered to protect your team from them. Often there will be too many obstacles to completely handle them all, meaning the player has to perform rapid threat assessment and triage under pressure. The player has access to a suite of software that can help them deal with various challenges, but may also

face hostile countermeasures that can disrupt them. Various actions both on the runner's level and the players can contribute to (or sometimes detract from) a global "heat" gauge that escalates the run's difficulty.

Management Layer: Using the money gained from their runs, the player will be able to hire and develop a permanent team of runners, each of which may have their own unique skills, strengths, weaknesses and quirks. The player will also be able to upgrade their cyberdeck with new programs and other enhancements.

3. Design Pillars

Triage over mastery. *The player should never clear everything perfectly. Success is smart prioritization and keeping cool under pressure.*

Attention as resource. *Complexity (mostly) comes from competing demands, not mechanical difficulty.*

Indirect control. *The player influences but never commands. The runners are people, not units.*

4. Mechanics

4.1 Progress Track / Signal Timeline

The core mechanic. Appears as a horizontal bar stretching across the screen, split into 5 horizontal sections, like a music staff. The runner team is represented by an icon on the left side in the middle track. Obstacles (such as doors, cameras and guards), objectives and other interactables (collectively called Signals) will usually appear on the left side and scroll toward the runner team's icon.

The horizontal distance of the track represents the runners' parallel distance from the Signal, while the vertical distance represents their perpendicular distance. A Signal that is on the same middle track as the runner team is something that can interact with the runner team once it has scrolled down the track to them. Most of the time, these are obstacles like cameras or doors and the runner team hitting them

will lead to some negative effect if the player hasn't disabled the Signal.

While the track is a straight path, it is meant to represent the overall path across many rooms and hallways that the team has to navigate, while the vertical separations represent side areas that can still potentially impact the run. For example, guards and drones are mobile obstacles that patrol. If they are in one of the upper or lower separations when they cross the team, then the team can pass by them without any negative effects. However, many things can affect the pathing of such obstacles, such as guards responding to alerts or the player using other signals to trigger a distraction.

Some Signals (usually doors) that appear in the perpendicular sections may offer boons such as alternate paths or paydata that increases their reward for the mission. The player must decide if it's worth risking the extra time and heat to pursue these side objectives.

By default, objects scrolling onto the timeline from the right will be about 30 seconds away from crossing paths with the runner team, though many things can affect this speed.

4.2 Mission Arcs

Most missions are divided up into smaller sections, such as: 1. Infiltration, 2. Objective, 3. Extraction. Each one has its own Signal Timeline, which may have special characteristics based on that section's framing. For example, an extraction section might have a faster scrolling timeline representing the runners having to move fast and the player having less time to respond to threats. Downtime between sections may allow the player to regroup and possibly allow the runners to use "camp" abilities a la Darkest Dungeon.

4.3 Heat

Heat represents how much attention the runner team and the player have drawn from corporate security. It grows at a steady rate, with many interactions raising or lowering it more. As it grows, difficulty scales higher across multiple vectors: additional obstacles spawn, mobile obstacles are more likely to cross paths with the runners, and

hacking becomes harder. Since most player actions have at least a small heat cost, many decisions the player makes will revolve around the heat cost versus the in-the-moment benefit.

4.4 Signal Scanning

It's not always immediately clear what a Signal represents or what its behavior might be. For a small time investment, the player can scan a signal continuously to gain more information about the signal. Information gained can include things like: what the Signal is (if not already revealed), hacking difficulty, patrol routes for some mobile Signals, any Intrusion Countermeasures (IC), which are software protections that interfere with the player in a variety of ways.

4.5 Hacking Minigames

To interact with a Signal, for example to disable a camera, the player will often have to complete a hacking minigame of various types (described in more detail in the Content section). These minigames are short puzzles that are meant to require attention without being too intellectually demanding. These minigames have several 'cadences' that affect how much and how often the player needs to interact with them. Some are simply "one and done" and once solved, the player is free to use any contextually available effects of that signal (Opening/Closing a door, triggering a distraction, etc.) Others have built in "process delays" that incentivize the player to deal with other things while waiting for the next part of the puzzle. Some minigames will be persistent, so the player's progress will be intact with each interaction, while others might "decay" or reset entirely if not solved in one session.

4.6 The Runner Team and Obstacles

The runner team are not simple lemmings and can solve most problems on their own -- but their solutions are by necessity often messier and riskier. Each obstacle has a range of effect, represented by a highlighted area projected from their Signal icon. When the runner team enters that range, there's an immediate and often scaling effect. For example, while the team is in a camera's range, extra heat is generated every tick. Once the runner team icon makes contact with the signal, the runners have to take matters into their own hands to

deal with the obstacle, which usually comes with a spike of heat gain, or some cost to the runners' health or stamina. The exact impact will be determined by behind-the-scenes dice rolls based on the runner's stats vs. the obstacles. The player's goal is not to disable every obstacle, but rather to minimize damage to the runner team so they can complete their objective and extract.

While the player has limited ability to command the team, they can ask the team to briefly slow down or speed up. This costs a small amount of heat per tick, but may allow the team to slip past guards or panning cameras without needing to hack them.

4.7 Software, RAM and Overclocking

The player's cyberdeck has access to several programs that can help them achieve their goals. Some of these programs are locked-in, while there are several (4-5?) open slots that allow the player some flexibility. Software programs take time to load and unload. While loading, unloading or active, they use up some amount of the player's RAM pool. When the RAM pool is depleted, the player can't deploy any more programs until they finish unloading some of the active programs.

The player can overclock their deck for brief periods to gain extra RAM, but doing so for more than a handful of seconds at a time will cause the system to reboot, cutting the player off from seeing or interacting with the Signal timeline.

Base programs include things like scanning Signals, expanding the Signal Timeline's horizon, or reducing heat. Other programs may have more specific niches, like making one specific type of puzzle easier, or causing some malus to a specific class of obstacle.

4.8 IC and White Hats

After the first few levels, the player begins to encounter intrusion countermeasures (IC or ICE). These are software programs that either passively improve an Obstacle, or trigger when the player attempts to hack it. They can cause a variety of disruptive effects to the player. These can include direct mechanical effects like adding modifiers to hacking minigames or causing Signals to be hidden until they are closer to the team, as well as meta-mechanical effects such as temporarily disrupting the players viewport, reversing their controls, etc.

Further on, the player can also encounter corporate White Hat hackers, which act similarly but are not attached to specific Signals and can instead disrupt the player at any point. (Concept to be further defined and expanded.)

4.9 Split Tracks and Set Pieces

On some missions, the runner team will be forced to split up to tackle separate objectives simultaneously. For example, the team's infiltrator may need to sneak his way through vents to open an unhackable door from the other side. This will result in brief sections where a second Signal Track appears and the player must split their time and attention between both. Some of these split tracks may have special mechanics based on the type of task. For example, there may be a "hold the line" sequence where one runner has to hold off enemies while the others continue toward the objective. That runner would split off on a second track where their icon is in the middle of the track and threats are approaching from both sides instead of the usual right-to-left flow.

There may also be "set-piece" encounters that offer slight twists on the usual mechanics and flow. For example, the runners may get in an extended fight with some corpo battle robot, and the player has to hack different subsystems through the fight. Or the runners are in an abandoned bio-lab, the lights went out and some genetically engineered creature is hunting them. The player has to hack doors and lights to block the creature and help the team navigate to an exit.

4.10 Management Layer

To be expanded. For now, consider a very stripped down version of Darkest Dungeon's roster management. The player can upgrade their runners' gear, buy consumables, improve their cyberdeck, buy new programs, and so on.

5. Content

5.1 Obstacle Types

Obstacles have two broad, binary classifications: they are either **stationary** or **mobile**, and they are either **mechanical** or **biological** (mech or meat, for short).

Stationary obstacles, such as doors and cameras, always stay in the same space in the Signal Timeline and always move toward the runner team at a fixed speed based on the speed the runner team is moving at.

Mobile obstacles, such as guards and drones, can move along the perpendicular axis, and can also sometimes move faster or slow toward the runner team depending on their patrol route or if they are responding to an alert or distraction.

Mechanical obstacles (cameras, doors, drones) can usually be seen directly interacted with by the player via the Signal timeline.

Biological obstacles (guards, mostly) are usually invisible to the player unless the player has hacked a nearby camera, or the runner team has spotted them. They cannot be interacted with directly, but they can be distracted or disabled by other nearby hackable objects.

5.1.1 Specific Obstacle Mechanics

Obstacle: Door [Stationary, Mechanical]

Description: The simplest obstacle. If locked and on the central path, the runner team will take time and gain heat as they find another way to pass it. Most likely to contain a time-intensive and/or multi-stage minigame.

Player Hacking Interactions: Open, Close, Lock, Unlock.

Obstacle: Trap [Stationary, Mechanical]

Description: Similar to doors, but have more dangerous effects if the runner team makes contact, often costing health or stamina.

Player Hacking Interactions: Disable, Enable, Overload (only on some traps; heat-intensive but can disable other nearby obstacles)

Obstacle: Camera [Stationary, Mechanical]

Description: A simple obstacle that can cause significant heat gain if the runner team enters its range. Usually has short and simple puzzles. Hacking a camera automatically reveals any nearby biological Signals (guards).

Player Hacking Interactions: Blind (quickly and easily disables camera, but at a higher base heat cost), Spoof (tougher minigame to hack, but no heat gain).

Possible Variations: Some cameras pan, and their line of effect shifts into perpendicular areas. With timing and luck, the runner team may be able to bypass them without needing to hack them. Some cameras, once disabled, will reboot themselves after a delay.

Obstacle: Drone [Mobile, Mechanical]

Description: Similar to cameras, but move along patrol routes that make it harder to predict if/when they the runner team will cross their path. Longer scans can reveal some information about their patrol route, such as how long until the move and which direction.

Player Hacking Interactions: Same as camera, plus Reroute (alters patrol pattern).

Possible Variations: Combat Drones that are more likely to damage the runners. Disruptor or Cloaker drones that hide nearby signals or disrupt the player's ability to hack.

Obstacle: Guard [Mobile, Biological]

Description: Similar to drones, but more reactive to alerts or suspicious activity. Invisible to the player unless near a hacked camera or very close to the runner team. Very likely to cost runner health or stamina if the team has to deal with them directly.

Player Hacking Interactions: None, but see Other Signals, below.

5.2 Other Signals

The signal timeline will have several other "internet of things" signals, representing a variety of connected objects. Many of these will be useless and only there to add visual interest and noise. Others can be hacked for a variety of effects, such as distracting or even disabling guards or drones. For example, the player might be able to hack a terminal near a guard to generate a false alert that the guard spends time investigating, allowing the runner team to slip by.

5.2 Minigame/Puzzle Types

Minigames trigger when the player tries to hack most things, and must be solved in order to access the contextual functions of that Signal. They present as movable popup windows, and organizing the player's interface will be a subtle part of gameplay. There will be

several types, with different types of Signals tending to have a specific minigame type, though this is not universal.
NOTE: This list is in “blue sky ideation” mode. Will likely be cut down to 3 or 4 minigames at base.

CORE Minigame: Terminal

Description: Less of a minigame and more of a systemic glue that helps sell the fantasy while requiring some level of skill and detail orientation. This is the front-end of every hacking interaction in the game -- when the player clicks on a Signal, it’ll pop up a terminal window and the player will usually have to do at least one small task before getting to the hacking proper.

Involves a simplistic shell command language with 4-5 commands, a few flags/modifiers, and some interaction with variables. It would play out as a simple call-and-response where the terminal presents some data and asks for input and you have to respond with the correct command to bypass it.

- Layer 1 would be matching the correct command.
- Layer 2 would be incorporating a variable (like some terminals might require you to run [ACCESS] <filename>, where <filename> is something you have to quickly scan the terminal output for. Maybe sometimes it’s [ACCESS] core.sys and sometimes it’s [ACCESS] system.dat or whatever)
- Layer 3 would be adding flags to respond to small details. Like some terminals maybe have a specific protection type and you have to do [ACCESS] core.sys -b to bypass it.

Should be just enough complexity that it feels pretty easy to remember but satisfying once you’ve internalized it.
Might also serve as a sort of core mini-mini game that interacts with other puzzle types (see example below).

Minigame: Decryption

Description: Longest type of puzzle by default, most common on doors. The player would be presented with a very simple cipher to decode, but the full key would not be immediately available. The player might be able to hurry the decryption with some terminal commands, or brute force it based on deduction.

For example, let’s say the cipher is ABCD=FGHI. The game shows you three pieces of info: 1) the cipher: ABCD, 2) the set of possible answers [E,F,G,H,I], and 3) two parts of the key: A = F, C = H. The player can wait for the rest of the key to complete, or can try and intuit the answer. Obviously this is a very simple version and there’s infinite room to scale complexity.

Minigame: Matrix-style Search
Description: You see a grid of shifting hexadecimal numbers. One static sequence is highlighted on the left (e.g., E9). You have to spot E9 in the shifting grid on the right and click it before it moves. Could be expanded to require some basic Hex math (which is pretty easy to learn by repetition, IMO), like you have to match two symbols that add up to a value, or you’re given one value and have to find the complementary symbol that adds up to a value.

Minigame: Memory Sequence
Description: The system presents a series of sequences of button presses and tones. The player has to remember the sequence and reinput it.

Minigame: Rhythm Section
Description: A brief guitar-hero style sequence of keystrokes moves toward a line (or similar such visual indicator), the player has to hit the right keys on the right timing.

Minigame: Connections
Description: Classic “draw a line from point A to point B without intersecting or crossing the danger zones.” Alternatively, a system where you have n inputs and n outputs. Signals fire from inputs in sequence. You're placing/rotating nodes in real-time to route each signal to its correct output before the next one fires. Complexity scales via speed, grid size, and obstacle nodes that block or redirect.

Minigame: Frequency Tuner
Description: The system presents a sine wave and you have to modify frequency and amplitude to match it.

Minigame: Layer Puzzle

Description: System presents an image made from multiple distinct layers, with each layer rotated to an incorrect angle. The player has to rotate the layers to create the correct image.

Minigame?: Background “Twister”

Description: A special type of minigame (or possibly an IC disruption) that requires the player to hold down one or more keys for an extended duration while continuing to manage other tasks. Alternatively, could be framed as a “timeout” where the player has to revisit a specific window and do some simple task to keep the connection from timing out, which might lead to heat gain or having to do a more complicated task to reestablish the connection.

5.3 Runner Archetypes

Protoype development will rely on generic runners without any special skills or features. To be developed, possibly to include specific classes that can fulfill different functions, especially with regard to split track sections (section 4.9).

5.4 Mission Types

To be developed. For now, consider standard Shadowrun/cyberpunk corporate espionage-type stuff. Sabotage, technology theft, headhunting, etc. Different missions will have different threat distributions, different mission arcs, etc.

6. Aesthetic Direction

Visual: Clean UI, high contrast, information-dense but readable. Think Hacknet meets Into the Breach. Most information can be communicated by simple shapes, colors and icons.
Audio: Synth tension, radio chatter, mechanical keyboard sounds, alert beeps. Audio cues for each obstacle type to help with information processing.
Tone: Cyberpunk noir with gallows humor. Corpo dystopia played straight but not grimdark.

7. Scope & Milestones

Priority	Features
Must have	Single track, 3-4 puzzle types, 5-6 obstacle types, core runner archetypes, basic resource system
Should have	Split track sections (one type), runner abilities, heat system
Nice to have	Multiple split types, crew management and characterization, procedural generation, “set piece” combat breakout system
Later	Full campaign, multiple corps, roguelike modifiers

8. Open Questions

- How does difficulty scale across a campaign?
- What happens when a runner dies? Is there permadeath?
- How much narrative framing does each mission need?
- How do players learn new puzzle types?

9. Cut Ideas (Graveyard)

Alt names/subtitles:
Bandwidth
Signal/Noise
Deniable Asset
RunTime
Time to Execute
Dead Channel
Signal::Loss
keepalive
Kill Process

11. Changelog