THE HERMIT MANUAL

An exhaustively long guide to everyone's least favorite spider

Written by Qrow

Accurate to patch v1.14.5.0, or whenever

Special thanks to authors of widgets needed for techniques in this document

Forenote

While numbers that are cited in this guide are all accurate and techniques listed should be able to be performed by any given player, connection and other factors may make things more difficult or downright impossible under given circumstances. Similarly, some opinions regarding balance and matchups in this guide are mine, so take them with a grain of salt.

Special addendum for this addition: Due to certain game and human limitations, widgets are needed for some techniques used in this mod. Approach at your own risk, as the legality of such widgets may be questionable at best.

Introduction

Welcome to the Hermit Manual! This guide details every aspect of the Hermit unit from the spider factory in quite possibly way too much detail. However, if you plan to solo Hermit (a viable strategy at the time of writing this guide) this should give you everything you need to know to have a relatively strong impact on team games. This guide mainly focuses on team games, but most information is applicable to 1v1 as well.

Calcs (aka the Code, Math and You: A Beginner's Guide)

A lot of the discussion in this guide involves a lot of math, so we need to talk about how Zero-K approaches unit fall damage and collision damage first in order to get some calculations down.

1. Unit-vs-Unit Collision

Handled by the DoCollisionDamage() function, triggered when weaponDefID == -3 and an attackerID is present.

Step 1, Speed check: Collision damage only activates if either unit's speed exceeds UNIT_UNIT_SPEED (5.5 elmos/frame). GitHub

Step 2, Relative speed: The relative velocity between the two units is calculated as:

```
relativeSpeed = sqrt((oVx-myVx)² + (oVy-myVy)² + (oVz-myVz)²)
```

Step 3, Damage per unit: Each unit's damage is calculated using LocalSpeedToDamage():

```
if relativeSpeed > velocityDamageThreshold (3):
    damage = (relativeSpeed - 3) × (mass × 0.6)
```

Buildings get a ×10 multiplier on their velocityDamageScale since they're treated as far more massive.

Step 4, Final damage dealt: The game deals the *lesser* of the two units' computed damages, multiplied by UNIT_UNIT_DAMAGE_FACTOR (0.8) so the smaller/lighter unit sets the damage ceiling. Both units take this same amount. GitHub

Step 5, Multipliers & immunity: The final damage is scaled by each unit's collisionDamageMult (set per-unit by other gadgets). Units can be immune (e.g. during jump/teleport), and allied units are protected from "no-ally-damage" flags.

Step 6, Velocity correction: After collision, both units' velocities are blended toward a shared momentum-conserving average (a weighted sum based on their masses), simulating an inelastic collision. GitHub

---

2. Unit-vs-Ground Collision (Fall Damage)

Triggered when weaponDefID == -2 with no attacker. This is the classic "fall damage."

Step 1, Decompose velocity into normal and tangent: The unit's velocity is split into a component **normal** to the terrain surface and a **tangential** component (sliding along it).

Step 2, Bounce physics: The normal component is reflected and scaled by elasticity (default 0.3), and the tangential component is scaled by friction (default 0.8), then an impulse is applied to simulate the bounce. GitHub

Step 3, Damage speed calculation:

damageSpeed = magnitude of (normal + tangent × TANGENT_DAMAGE)

where TANGENT_DAMAGE = 0.5, so sliding impacts count for half.

Step 4, Convert speed to damage (same LocalSpeedToDamage() as above):

```
if damageSpeed > 3:
    damage = (damageSpeed - 3) × (mass × 0.6)
```

Step 5, Out-of-map bonus damage: If the unit is outside the map boundaries, additional damage is added proportional to how far out of bounds the unit is, scaled by unit.health / 800. GitHub

Step 6, Multiplier applied: Final damage × collisionDamageMult[unitID] (or 1 if unset).

---

3. Unit-vs-Wreck/Feature Collision

Triggered when weaponDefID == -3 and attackerID == nil (no unit attacker — a feature like debris).

The engine's native damage value is taken and multiplied by DEBRIS_SPRING_DAMAGE_MULTIPLIER = 10 (described in the code as "tweaked arbitrarily"), then scaled by collisionDamageMult. The unit also gets a velocity-scaling impulse of 0.3 (i.e. it loses 70% of its speed). GitHub

> Note: Feature collisions still use the native engine value as a base, unlike unit-unit and ground collisions which are calculated entirely from scratch.

Key Constants

Constant	Value	Purpose
UNIT_UNIT_SPEED	5.5	Minimum speed to trigger unit-unit damage
UNIT_UNIT_DAMAGE_FACTOR	0.8	Scales final unit-unit damage down
velocityDamageThreshold	3	Speed floor before any damage is dealt
velocityDamageScale	mass × 0.6	How hard a hit converts to damage
TANGENT_DAMAGE	0.5	Sliding/grazing impact weight
elasticity (default)	0.3	How much units bounce off the ground
friction (default)	0.8	How much lateral speed is kept on impact
DEBRIS_SPRING_DAMAGE_MULTIPLIER	10	Wreck collision damage multiplier

---

- Zero-K ignores the engine's native input for fall and unit collisions entirely and calculates damage from scratch, but the gadget-calculated value still ignores armor (so a unit with armorMult = 0 can still take collision damage). Feature collisions are separate and still use the native engine value. GitHub
- Units can be granted temporary fall damage immunity (e.g. during a jump) via SetUnitFallDamageImmunity.

Stats

Listed below are the relevant stats as of time of writing.

Stats	
Cost	145

Hit Points	1550
Mass	185
Movement Speed (elmo/s)	54
Turn Rate (deg/s)	316
Vision Radius (elmo)	420
Transportable	Light

Abilities

Improved Regen	
Idle Regen (HP/s)	10
Time to enable (s)	10

Weapons

Light Plasma Cannon	
Damage	141
Reload Time (s)	2.6
Damage per Second	54
Range (elmo)	350
Area of Effect (elmo)	18
Projectile Speed (elmo/s)	280

If you're new here, Hermit is a "spider assault", which really just means you can spam the hell out of it and get a lot of value in most scenarios. Hermit mainly excels at this job due to a few factors:

- Big chonky HP pool (1500 is a lot!) That allows you to walk down most factories with large numbers
- Great autoregen that helps out a lot (10hp/s after 10s)
- Being a spider means you can retreat to advantageous positions very easily
- You beat most other assault at their own game, and even skirmishers due to decent move speed and good firing angles
- With micro or widgets, you can use hermits as long range artillery, make Jugglenaut into a REAL strider, pond skip units out of bounds at light speed and clip through terrain (and live) - the possibilities are endless!

Build Order/Early Game Strategy

Early game is immensely important when attempting to do the main hermit strategy, which is 66% autoretreat fight-move (henceforward referred to as 66). When considering your build order, it is important to consider the following:

- How fast am I going to be approached by my opponent? Small numbers of hermit don't fare too well against micro'd raiders in the early game.
- How much E will I need to prepare for repair stations?
- Do I need to focus on expansion or units first for taking ground on smaller maps?

**Protip! Keep a calculator by your desk for doing E and damage calculations if you aren't good at math.*

Starting Queue

Usually, your starting queue looks something like this:

s. denotes starting to build a unit (placing nanoframe). *spc. denotes placing an item in the middle of your existing queue with space + left click.* *r. denotes resume.*

[Token]Factory -> s.Solar1 (wind if better) -> spc.4xFlea~loop.Hermit -> r.Solar1 -> Solar2 -> Mex -> Storage -> Caretaker...*

**Remember to set 66% autoretreat post Flea.*

Usually you will want to get a builder in there at some point.

Generally, a good rule is that you want to get about 5 hermits up before using them to hold ground, as smaller numbers tend to still get trounced by microed raiders before making it to a repair station.

Caretaker Placement, and E and You

Besides Hermits, the other most important thing you will need to consider building are Caretakers, and by extension building repair stations.

Caretakers are 10BP, and consume 10E/s for a total duration of a Hermit repair. Auto-regen is free, but caretakers do consume E, so having an appropriate supply for repairs and also for using your metal is very important. Consider capping an early Geothermal or building a fusion in order to secure your E needs.

Repair Stations

We mentioned repair stations earlier, but what do repair stations actually need?

Anywhere on the map can be set as a retreat zone, which will be denoted for *you only* as a red circle with a wrench on the map. Other players cannot see your personal retreat zones, so communicate accordingly.

Repair stations usually need 3 things:

- BP source for repairs

- Some source of protection while repairing (from Air/etc, usu. in form of terrain)

- Some way to ensure that the area does not murder hermits on the way out (i.e. don't put retreat zones in chokepoints)

Following these three rules will generally ensure your hermits live as long as possible.

On flat maps, terrain where it is convenient to place a RS may be in short supply, so terraforming a hollow box and putting stuff inside may be useful, and also gives you privacy from your enemies' radar.

Other Units

Spider Factory

Venom

Useful to deal with some raiders, but rarely used otherwise. Mass hermit fairs pretty well against raiders already. Occasionally useful when you need things to not move, although Widow also does this job.

Redback

Useful for anti-raider or anti-flea properties in the mirror match. Not used a whole lot otherwise.

Recluse

Solves some problems in awkward spots you might find yourself in regarding range. Usually quite fragile so they have to be micro'd separately from your Hermit wall. Worth considering in the mirror, sometimes.

Widow

Useful for stunning certain heavies that are annoying, or tempo fixing when the enemy somehow conjures up a crab from nowhere. Also useful as a way to work around DDMs, which are quite strong against hermit.

Crab

If you lose tempo to build this in the mirror, it is losing. If you can get this for free from reclaim while also keeping hermit tempo, it's very good in the mirror.

Useful outside the mirror in all sorts of situations where you need a bit more punch to your offensive, or for holding ground.

Other Compositions

(Listed as compositions here and not individual units or factories)

Scapel/Dagger/Lance

Combining Hermit with the hover factory gets you a lot of good options in terms of DPS projection and threat removal before the majority of the hermits have to walk up to the target. By moving behind a hermit wall, a player can get a lot of value out of the range of Scalpel, and also help hermits by providing daggers for raider management and Lance for threat removal.

Sling

Useful for combining with hermit balls in order to either force your opponent to clear out of an area or for forcing them to fight you when you have superior numbers. Also very cheap and good.

Iris

Walking a cloaked Hermit ball into your opponent's base generally wins the game. Also useful for dealing with some tricky artillery or sniper situations.

Thug/Iris/Rogue

Provides some more beef to the hermit wall and also boosts its protection versus things like planes. Rogue offers some decent damage projection in the same way that Scalpel does. Usually quite expensive.

Badger/Impaler

Useful for the same reason as sling, but at more of a cost. Badger has the slight upside of being quite mobile, so it can effectively help retreating hermits more.

Juggle

See most of the entire tech section later.

Newton

Vitally important to late game and tech based solutions for problems. Being able to flash build these is valuable. If only we had Newton with legs.

Placeholder

You sometimes need to placeholder your own hermits for certain tech. Also helps against tanks.

Lobster

Also needed for certain loop-recycle tech.

Tech

Easy (Essential) Tech

this is where it all begins

Retreat Zoning

Don't like it? HIT DA BRICKS

This isn't really tech, but putting your Hermits on 66% autoretreat and having safe areas for them to back up to is by far the easiest "tech" you need to know with them. Just doing this makes them much stronger and more annoying to deal with for the enemy.

Warpgate

Go Places Fast!

By using a newton turret and a low angle ramp, you can effectively zip hermits to anywhere on a map! Simply build a newton turret, push your hermit and set an auto retreat zone where they land. Hermits have free autoregen, so if you have some excess E or just some time for them to heal, this can get them around much faster.

If you want even faster travel, you can set up a warpgate setup with pull newtons at the end that will slow the hermit down enough for it to land safely. Consult the math at the beginning of this guide for appropriate calculations.

Puck Shot

Throw a rock at em'

Because hermits have so much HP, a primary use for hermits becomes using them as a way to abuse Zero-K's unit collision damage. We will explore this more later, but for now the easiest way to do damage with hermits is simply to impulse them into enemy structures or units with a newton. Remember - the damage ceiling is set by the smaller unit!

This tech can be easily aimed with a small newton array and ramp.

Intermediate Level

Pond Skipping

For when you need some extra bounce in your step

By throwing hermits at a low enough angle to the ground, you can abuse the retrograde impulse applied to terrain collision and effectively skip hermits across the map at a low cost to their HP.

For this on average, you will need:

* an extremely low angle ramp

* 2 newtons

Try playing around with it until you get the hang of it, it's not too hard.

Out of Bounds Skip-Clipping (a.k.a. Watari)

Spiders through space-time

For this tech, we need to understand a few things about how out of bounds units work in Zero-K, and how they are repositioned when they are detected as out of bounds.

Damage is applied via a OOB modifier as discussed earlier. Generally the math considers:

- The unit itself
- How far OOB you were

- Some speed math

Units are generally repositioned on the map in a straight path to the border if they do not die, as often seen with wrecks.

By abusing the earlier skip tech to barely skip hermits off the map, we can use the skip angle to bypass certain terrain features such as hills or other pesky things at the map edge in order to pass out hermits up the map and have them skip the obstacle. Because of the nature of how skipping works, besides the OOB damage penalty, hermits do not take much damage for doing this, and you can use it to scout or hit certain hard to reach targets.

Spinning Top (a.k.a. Wiggle)

SPEEEN *widget only

This tech actually isn't very hard, but due to the fact you need an autopress widget to pull it off and the fact that it is quite inconsistent even if you do have the widget, I'm putting it here essentially just as a sidenote.

If you have hermits trapped in a placeholder ball, you can abuse momentum and unit collision by rapidly (beyond human clicking) trying to move the hermits in a random direction. If successful, the hermits will turn upside down in the placeholder ball, collide, and bounce out (taking some damage on the way). I do not know why this works or sometimes doesn't work, please don't ask.

REAL Tech (Hard)

Tech in this section is usually extremely case specific and in some cases very finicky. Many require high APM or custom widgets you need to develop yourself. Proceed at your own risk.

Recycle-Bounce Damage Negation Lobster Loops (RBDNLLs)

What are these names anymore

By throwing a hermit into an enemy unit and just in time after applying damage lobstering your units back, you can apply a state of damage invuln to your own hermits upon being thrown by lobster. This takes advantage of the fall damage

negation trigger that hermit applies. The command order goes as follows:

- Impulse hermit
- Hit Target
- Immediately Lobster Back
- Heal, Recycle, Loop tech
- Profit

This tech, if done effectively, essentially can let you transform your frontline hermits into massive amounts of burst damage if done and micro'd well.

Nebula

Isn't that another unit

With custom widgets, you can use Jugglenauts to juggle (push pull) hermits in an orbit around the jugglenaut, thus creating a constant ring of collision damage around the Juggle to wipe out pesky raiders and such.

Figure out the math on this one yourself.

PoolShot

For when placeholders need to die

By shooting your own hermit at a ball of your hermits in a placeholder ball, you can effectively kick all of them out of being placeheld and at the same time turn your hermits into living grapeshot! This one is easy to understand, but the aiming is hard.