



THE ASTER MISSION

HIGH ORBIT

STUDY GUIDE

Bring the stars back to life.

FOR

The engineer (that is you)

PLAYING TIME

25 to 45 minutes

REPAIRS

Eight, in order

SUGGESTED AGES

10 to 14

Contents

01	Welcome aboard Aster
02	How a repair works
03	The skill this game builds: troubleshooting
04	Repair 1 · Power relay
05	Repair 2 · Garden rover
06	Repair 3 · Coolant drive
07	Repair 4 · Energy console
08	Repair 5 · Airlock interlocks
09	Repair 6 · Solar array
10	Repair 7 · Launch sequencer
11	Repair 8 · Orbital navigation
12	Replaying and improving your score
13	Glossary

This guide never gives you an answer. It tells you what each repair asks, what the controls do, and what to think about before and after you play. The solving is yours.

Welcome aboard Aster

Somewhere beyond Earth, a little station carries a garden meant for a new world. Then the lights go out.

HIGH ORBIT is a story-driven repair game. You play the engineer who wakes up aboard **Aster**, an orbital seed station, after an impact has knocked its systems offline. Aster is not an abandoned wreck. It is carrying a seed garden to the colony of **New Meridian**, and the seedlings are its smallest passengers. Your job is to bring the station back to life, one system at a time, and then send a signal home.

You are not alone. **PIP**, the station's small intelligence, is awake, a little dazed, and very glad to see you. PIP introduces each repair, explains what is wrong, and says a short line when you succeed. The game is cosy and hopeful. There is no countdown, no failure limit and nothing that can go permanently wrong. Every failed test is simply information you can use.

Your role

You are the engineer. Each of the eight repairs is a puzzle that stands for a real kind of thinking: following a circuit, writing a sequence of instructions, balancing a machine, sharing a budget, combining logical conditions, predicting side effects, scheduling work and planning a route within limits. The game calls these skills by name at the top of every repair screen, for example *Circuits · follow the connection* or *Logic · combine conditions*.

The eight repairs

#	SYSTEM	CHAPTER TITLE	SKILL
1	Power relay	A little light	Circuits · follow the connection
2	Garden rover	Something worth saving	Code · think in sequences
3	Coolant drive	Keep the heart turning	Mechanics · balance the system
4	Energy console	Enough for everyone	Resources · plan within limits
5	Airlock interlocks	The language of locks	Logic · combine conditions
6	Solar array	A ripple through the array	Patterns · reason about side effects
7	Launch sequencer	Everything in its own time	Scheduling · coordinate dependencies
8	Orbital navigation	The long way home	Navigation · optimise a constrained route

The repairs unlock in order. You cannot open repair 3 until repairs 1 and 2 are restored. They get harder as you go: the first is an introduction, the last is called the final challenge.

Finding your way around

- **Splash screen:** the logo and tagline. Choose *Enter*.
- **Main menu:** *Begin mission* (or *Continue mission* if you have played before), *Mission records*, *Settings*, and *New mission* once a mission is under way.
- **The cabin:** the illustrated station with eight glowing markers, one per repair. The next repair is highlighted. A dock of icons along the bottom does the same job, and PIP's current line tells you what to do next.
- **Repair screen:** the puzzle itself (see the next section).

- **Mission records:** your XP, stars and time for every repair, and your rank.
- **Settings:** sound effects, ambient motion, fullscreen and keyboard guidance.

Press **Escape** to leave a repair and return to the cabin, or to open the menu from the cabin. Everything can be done with a mouse, touch, or the keyboard (Tab to move between controls, Enter to press).

PIP

"Oh. Hello, engineer. I'm PIP. Most of me is still asleep. Could you start with the power relay? I'd quite like to see the stars again."

How a repair works

Every repair follows the same rhythm: open it, read the brief, work the controls, test, and read the feedback.

1. Open the repair

Select the glowing marker in the cabin or the matching icon in the dock. The repair opens full screen with a short objective at the top, for example *Rotate the connectors. Link SOURCE to EXIT.*

2. Read the brief

Under the objective is a short description of the problem. The essentials you need are always on screen. If you want more, open **How it works**, a collapsed panel that explains the rules in full. You never need it to finish, but it is there if you want it.

3. Use the controls

Each repair has its own controls: tiles to rotate, command buttons, a gear selector and slider, plus and minus buttons for energy cells, switches, a grid of lights, start-time menus or a route map. Every change you make is counted as a **move**. Reset buttons put the puzzle back to the start if you want a clean slate.

4. Test

When you think you have it, press the test button (it has a different name in each repair: *Test connection, Run program, Engage the pump, Test allocation, Release the airlock, Synchronise array, Validate launch sequence, Commit the course*). The game checks your work. If it fails, the feedback tells you what is wrong, often quite precisely. If it passes, the system is restored.

5. Hints

Every repair has three hints, each more specific than the last. Press *Get a hint* once for a nudge, twice for a method, three times for the answer. Hints cost XP, so try the puzzle first. Re-reading a hint you have already opened costs nothing extra.

6. Telemetry

The **Engineer's telemetry** panel shows your live **moves**, **tests** and **time** for the repair, the move target, and the star rules. Time is recorded because engineers like to know, but it never affects your score.

7. Stars and XP

HOW EACH REPAIR IS SCORED

You start with **1000 XP** for the repair. Then:

- minus **75** for each failed test;
- minus **100** for each hint you reveal;
- minus **5** for each move over the move target.

You never score less than **200**, and time is never deducted.

★★★ no hints and at most 1 failed test

★★ at most 1 hint and at most 3 failed tests

★ anything else. The repair still counts as restored.

REPAIR	MOVE TARGET	REPAIR	MOVE TARGET
1 Power relay	11	5 Airlock interlocks	3
2 Garden rover	8	6 Solar array	5
3 Coolant drive	3	7 Launch sequencer	4
4 Energy console	12	8 Orbital navigation	5

Your XP adds up across the mission to give you a rank: **Cadet**, then **Technician** at 2000 XP, **Flight engineer** at 4000 and **Master engineer** at 6000. A perfect mission is 8000 XP and 24 stars.

Saving

Your progress saves on the device you are using, so you can close the game and come back. If you are playing inside the edu.games player, your progress lasts for the page session only, so try to finish in one sitting or note where you got to. *New mission* wipes everything and starts again; the game asks you to confirm first.

The skill this game builds: troubleshooting

Every repair in HIGH ORBIT is a troubleshooting loop. Learn the loop and the puzzles become much friendlier.

1. PREDICT

Before you press anything, say what you expect to happen. "If I rotate this tile, the path should turn down." "If I pick the 24-tooth gear, the speed should drop." A prediction gives your test something to be compared with.

2. TEST

Make the change and press the test button. Testing is not cheating and it is not giving up. It is how engineers find out. A failed test costs a little XP, but it is also the fastest source of information in the game.

3. READ THE FEEDBACK

The game tells you what went wrong: which step was blocked, which rule failed, which time slot is overloaded, how many checks passed. Read it slowly. The first wrong thing is usually the thing to fix.

4. CHANGE ONE THING

Fix one thing, then test again. If you change three things at once and the test passes, you do not know which change mattered. If it fails, you do not know which change broke it.

Habits that help

- **Work backwards from the rules that leave no choice.** Some rules fix a value outright (a switch that must be on, a gear that must give one speed). Settle those first and the rest of the puzzle shrinks.
- **Write things down.** A pencil sketch of the grid, the route or the time slots is often faster than holding it in your head. That is what the notes boxes in this guide are for.
- **Use Reset when the state is confusing.** Starting again from a known position is a real engineering technique, not a defeat.
- **Count your moves, but do not obsess.** The move target rewards planning. Five XP per extra move is small; a wasted hint is 100.
- **Ask for a hint when you are stuck, not when you are bored.** If you have made a prediction, tested it and read the feedback twice and still cannot see it, that is a good time.

PIP, on the garden rover

"An algorithm is a series of clear steps. Test it, find the first wrong step, then revise."

Power relay

Chapter: A little light

The situation

Aster is on emergency power. PIP says most of it is still asleep. The relay that feeds the cabin lights is intact, but the impact rotated its connectors, so power from the source cannot reach the exit. Objective on screen: *Rotate the connectors. Link SOURCE to EXIT.*

The game's brief: *The impact twisted the relay connectors. Turn the tiles to carry power from the source to the cabin lights.*

The controls

- A 3×3 board of connector tiles. **Select a tile to rotate it 90° clockwise.** Each tile has two openings; rotating moves both.
- Power enters at the **left edge of the top-left tile (SOURCE)** and must leave at the **right edge of the bottom-right tile (EXIT)**.
- Lit connectors have power. As you rotate, you can see how far the power reaches.
- *Test connection* checks the circuit. *Reset tiles* returns every tile to its starting angle.



The nine tiles, numbered left to right, top to bottom. Source enters tile 1 from the left; exit leaves tile 9 to the right.

Key terms

Circuit: a complete, unbroken path for electricity. **Source:** where power comes from. **Connector:** a piece that joins two neighbours. **Continuous path:** a route with no gaps.

BEFORE YOU PLAY

1. A tile has two openings. How many different positions can it have after rotating? What happens on the fourth rotation?
2. Power can only flow between two tiles if both have an opening facing each other. Why is one opening not enough?
3. Look at the lit tiles. What does the first unlit tile along the path tell you?

AFTER YOU PLAY

1. Which tile did you have to rotate most times? Could you have planned it so you rotated it fewer times?
2. The move target is 11. How close were you, and where did extra moves come from?
3. PIP says a circuit needs an unbroken path from source to destination. Where have you seen the same idea outside the game?

Notes

The situation

The lights are on, and PIP reveals what Aster is really carrying: a seed garden for the next colony. The seedlings are thirsty. A small garden rover is stranded beside the seed vault, and it needs a program to deliver water. Objective on screen: *Program a route to the seedling. Avoid the crates.*

The game's brief: *The seedlings are thirsty. Program the little rover to deliver water to the garden. It starts at the arrow, facing right.*

The controls

- A 5×4 grid. The rover starts in the top-left cell, facing right. The sprout is the goal. Striped crates block the way.
- Three command buttons: **Forward** (move one square in the direction you are facing), **Turn left** and **Turn right** (turn in place without moving).
- Your program appears as a queue of commands. Select a queued command to remove it. The program can hold **up to 12 commands**.
- *Run program* animates the rover step by step. *Clear* empties the queue.
- If a step hits a crate or the edge, the feedback tells you which step was blocked.



The garden floor as the game shows it: striped cells are crates, the sprout is the goal.

Key terms

Algorithm: a series of clear steps. **Sequence:** the order the steps run in. **Program:** a list of commands a machine follows exactly. **Debugging:** finding the first wrong step and fixing it.

BEFORE YOU PLAY

1. Trace the only open path from the rover to the sprout with your finger. How many squares does it move, and how many times does it change direction?
2. A turn does not move the rover. If the rover faces right and you want it to go down, is that a left turn or a right turn?
3. What is the smallest number of commands that could possibly work? Why is 12 the limit?

AFTER YOU PLAY

1. If your first run failed, which step was blocked and what had gone wrong just before it?
2. Did you run the whole program at once, or build it in pieces? Which is better for finding mistakes?
3. Real robots follow instructions exactly, even silly ones. What does that mean for how you write instructions?

Notes: write your program here before you type it



Coolant drive

Chapter: Keep the heart turning

The situation

The seedlings are drinking again. Now a rattling coolant pump is running too fast to keep the station cool. You need to slow it with the right driven gear and then set the coolant flow so the temperature is just right. Objective on screen: *Tune the pump to 30 RPM and 45°C.*

The game's brief: *The pump is spinning too quickly to keep the station cool. Fit the right driven gear, then set the coolant flow.*

The controls

- The **driver gear** is fixed: **12 teeth, 90 RPM.**
- Choose a **driven gear**: 18, 24 or 36 teeth.
- A **coolant flow** slider from 0 to 5.
- A live readout shows the speed and temperature your settings produce.
- *Engage the pump* tests the settings.

THE TWO RULES, AS THE GAME STATES THEM

Speed = $90 \times 12 \div \text{driven teeth}$

Temperature = $60 - (5 \times \text{flow})$

Target speed 30 RPM. Target temperature 45°C.

Key terms

Gear ratio: the relationship between the teeth on two meshed gears. **RPM:** revolutions per minute. **Driver / driven:** the gear that is turned by the motor, and the gear it turns. **Inversely proportional:** when one goes up, the other goes down by the same factor. Meshed gears turn in opposite directions.

BEFORE YOU PLAY

1. Before touching anything, work out the speed for each of the three gears using the formula. Which one hits 30 RPM?
2. The temperature only depends on flow. What flow gives 45°C? What does flow 0 give?
3. The move target is 3. What are the three moves?

AFTER YOU PLAY

1. Why does a bigger driven gear turn more slowly than the driver?
2. The two settings do not affect each other in this puzzle. How did that make the problem easier than it first looked?
3. Where else do gear ratios show up? Think about bicycles.

Notes: your working

Energy console

Chapter: Enough for everyone

The situation

No rattle, no alarms, just a heartbeat. Aster has exactly twelve energy cells left and four essential systems: **Habitat**, **Seed garden**, **Navigation** and **Comms beacon**. The labels on the console are damaged, but four relationships survived. Objective on screen: *Share all 12 cells. Satisfy the four relationships.*

The game's brief: *The labels on the energy console are damaged. Use the surviving relationships to share exactly twelve cells across four systems.*

The controls

- Four sliders, one per system. A readout shows how many of the 12 cells you have used.
- *Test allocation* checks the four relationships. *Reset* clears every slider.

THE FOUR RELATIONSHIPS, AS THE GAME STATES THEM

- Navigation needs at least 1 cell.
- Habitat needs exactly 1 more than navigation.
- The garden needs exactly twice navigation.
- The beacon needs the same as habitat.

Use all 12 cells. Every relationship must hold at the same time.

Key terms

Budget: a fixed total you must not exceed, and here must use exactly. **Constraint:** a rule your answer must obey.

Linked equations: several rules that share the same unknowns, so they must be solved together. **Unknown:** a value you do not know yet, often given a letter.

BEFORE YOU PLAY

1. Three of the four rules describe a system in terms of navigation. If navigation is N , write the other three systems in terms of N .
2. Add them up. What expression gives the total number of cells? What must it equal?
3. Why does solving for navigation first make the whole puzzle fall into place?

AFTER YOU PLAY

1. If the station had 17 cells instead of 12, would the same method work? Would there be a whole-number answer?
2. Did you solve it on paper or by sliding and testing? Which took fewer moves?
3. PIP says a budget is a constraint: meet essentials first, then check the total. Where do people do this in real life?

Notes: let navigation = N



The situation

The essentials are safe, and PIP warns that the harder part begins. The impact locked the outer hatch, and the hatch is listening to five conflicting safety signals. Objective on screen: *Pass all six interlocks with at most three switches on.*

The game's brief: *The hatch is listening to five conflicting safety signals. Set switches A-E so that every interlock passes, using at most three active switches.*

The controls

- Five switches, **A to E**. Select a switch to toggle it ON or OFF.
- A list of six checks that update live, so you can see which pass and which fail.
- *Release the airlock* tests all six at once. The feedback tells you how many passed. *Reset* turns everything off.

CHECK	MEANING
A XOR B	Exactly one of A and B is on.
B OR D	At least one of B and D is on.
C XOR D	Exactly one of C and D is on.
NOT (D AND E)	D and E are not both on.
E must be ON	E is on.
At most 3 switches ON	No more than three of the five are on.

Key terms

AND: true only when both are true. **OR:** true when at least one is true. **XOR** (exclusive or): true when exactly one is true. **NOT:** reverses a condition. **Interlock:** a safety rule that stops a machine operating unless conditions are met. **Boolean:** a value that is either true or false, on or off.

BEFORE YOU PLAY

1. One rule names a single switch and gives you no choice. Which is it, and what does it force?
2. Once you know that switch, look at NOT (D AND E). What does it now tell you about D?
3. Follow the chain: with D decided, what does C XOR D say about C? How many switches have you used so far?

AFTER YOU PLAY

1. How many different ways can five switches be set? (Each has two states.) Why is trying them all a poor plan when reasoning takes four steps?
2. What is the difference between OR and XOR? Give an everyday example of each.
3. Why would a real airlock use interlocks like these instead of a single big button?

Notes: A ____ B ____ C ____ D ____ E ____

Solar array

Chapter: A ripple through the array

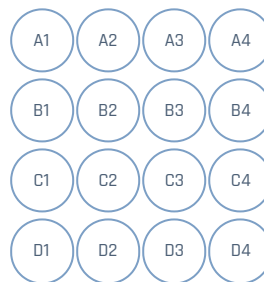
The situation

The airlock recognises you. Outside, sixteen solar collectors are pointing into shadow, and their controls are linked: changing one affects its neighbours. Objective on screen: *Light all 16 collectors. Every press flips a cross.*

The game's brief: *Bring all sixteen collectors into sunlight. Each control flips itself and its immediate neighbours: lit becomes dark, and dark becomes lit.*

The controls

- A 4×4 grid of collectors, rows A to D and columns 1 to 4. Select any cell to flip it **and** the cells directly above, below, left and right. Diagonals do not change. Edge and corner cells simply have fewer neighbours.
- A readout shows how many collectors are lit.
- *Synchronise array* tests whether all sixteen are lit. *Reset array* returns to the starting pattern.
- There is no move limit; the move target of 5 is a guide to what a planned solution looks like.



How the cells are named. Pressing B2 flips B2, A2, C2, B1 and B3.

Key terms

Side effect: a change caused somewhere other than where you acted. **Toggle:** a switch that flips between two states.

Cancel: two identical toggles undo each other. **Parity:** whether a count is odd or even; a cell ends up flipped if it was flipped an odd number of times.

BEFORE YOU PLAY

1. Pick a cell and, before pressing, name every cell that will change. Press it. Were you right?
2. Press the same cell again. What happens? What does that tell you about pressing any cell twice?
3. Does the order you press cells in matter? Try two cells in both orders and compare.

AFTER YOU PLAY

1. Why is “fix the dark ones one by one” a trap in this puzzle?
2. If order does not matter and pressing twice cancels, what is the only thing that really matters about a solution?
3. PIP says a local action can change several parts of a system. Where have you seen that happen (in code, in a team, at home)?

Notes: sketch the grid and mark the cells you plan to press



Launch sequencer

Chapter: Everything in its own time

The situation

Sunlight at last. Aster has the power to leave, but the launch systems share one power bus. The right jobs must run in the right order, without overloading it. Objective on screen: *Schedule six jobs. Finish by time 6. Stay within 5 power.*

The game's brief: *Fit six departure jobs into six time slots. Jobs run continuously and share a five-unit power bus. A dependency must finish before the next job begins.*

The controls

- Six jobs, each with a start-slot menu (slots 0 to 5). A job starting at 0 with duration 2 runs in slots 0 and 1 and finishes at 2.
- A timeline shows the power used in each slot out of 5; an overloaded slot is highlighted.
- *Validate launch sequence* checks the plan; the feedback lists the first few conflicts. *Reset* puts every job back at slot 0.

JOB	DURATION	POWER	MUST RUN AFTER
A · Air pressure	2 slots	3	no prerequisites
B · Navigation boot	2 slots	2	no prerequisites
C · Seal test	1 slot	2	after A
D · Engine prime	2 slots	3	after B
E · Launch handshake	1 slot	2	after C & D
F · Final ignition	1 slot	4	after E

Key terms

Dependency: a job that must finish before another can start. **Capacity:** the most the shared resource can supply at once (here 5 power). **Schedule:** a start time for every job. **Deadline:** everything must finish by time 6. **Overlap:** jobs running in the same slot, allowed only if the power adds up to 5 or less.

BEFORE YOU PLAY

1. Add up the durations. If the jobs ran one after another, how long would they take? Why must some overlap?
2. Which jobs have no prerequisites? What is their combined power? Can they share a slot?
3. Follow the longest chain of dependencies from start to F. How many slots does that chain need at minimum?

AFTER YOU PLAY

1. PIP says dependencies determine order and capacity determines what can run together. Which of the two caused more of your failed tests?
2. Is there any slack in your schedule, or is every slot needed?
3. Where do people schedule like this in real life (cooking a meal, a building site, a computer running programs)?

Notes: slots 0 · 1 · 2 · 3 · 4 · 5



Orbital navigation

Chapter: The long way home

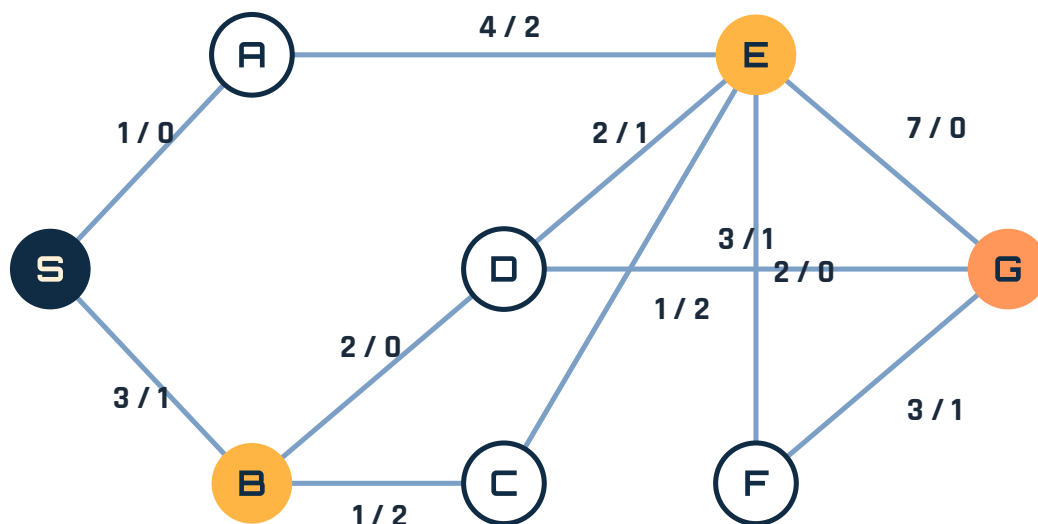
The situation

Departure checks are complete. Home is beyond a debris field, and the ship needs two relay keys on the way: the navigation key at **B** and the beacon key at **E**. PIP warns against the shortest-looking wrong turn. Objective on screen: *Reach G via B and E. Budget: 12 fuel, 3 radiation.*

The game's brief: *Plot a course from Aster (S) to home (G). Collect the navigation key at B and beacon key at E. The ship can spend at most 12 fuel and tolerate at most 3 radiation.*

The controls

- A map of eight waypoints, S, A, B, C, D, E, F and G, joined by legs. Each leg is labelled **fuel / radiation**. Every leg can be flown in either direction, and flying a leg again costs again.
- Build the route one leg at a time by selecting a connected destination from the choices below the map. A readout shows your fuel and radiation so far.
- *Undo leg* removes the last leg at no cost. *Reset route* starts again from S. *Commit the course* tests the finished route at G.



The route network as the game draws it. Labels are fuel / radiation. Amber waypoints hold the two keys; G is home.

Key terms

Graph: a set of points (nodes) joined by lines (edges). **Constraint:** a limit the route must respect (fuel at most 12, radiation at most 3). **Trade-off:** a cheaper leg in one resource may be dearer in another. **Valid route:** one that visits both keys, ends at G, and breaks neither limit.

BEFORE YOU PLAY

1. Which legs have 2 radiation? With a budget of 3, how many of those can you afford on the whole trip?
2. Find the cheapest-fuel way from S to B, and from E to G. Are the cheap legs also the safe ones?
3. You must pass through B and E. Does the order (B first or E first) matter for the budget?

AFTER YOU PLAY

1. PIP says a valid route must satisfy every constraint, not just minimise distance. Which constraint turned out to be the tight one?
2. Was there a leg that looked tempting but broke a limit? Which, and why?
3. Satnavs, delivery drivers and spacecraft all solve routes with limits. What limits do they juggle?

Notes: route and running totals

Replaying and improving your score

A restored repair shows a record screen rather than a replay, so the way to improve is to start a new mission and play it better.

What a better score looks like

- **Fewer failed tests.** Predict before you test. One failed test still allows three stars; two drops you to two stars at best.
- **No hints.** Any hint rules out three stars for that repair. A single hint still allows two stars.
- **Moves near the target.** Every move over the target costs 5 XP. Planning on paper first (the rover program, the five array cells, the six start slots, the route) is the easiest way to hit the target.
- **Time does not matter.** Take as long as you like. There is no bonus for speed and no penalty for thinking.

Reading your mission records

Open *Mission records* from the main menu. For each repair you will see XP, stars and active time. Look for the repair with the lowest XP: that is where a replay will gain the most. Your rank at the top is the sum of all your XP.

RANK	TOTAL XP NEEDED	ROUGHLY MEANS
Cadet	0	Mission under way
Technician	2000	A few repairs, or several at one star
Flight engineer	4000	Most repairs with solid scores
Master engineer	6000	Eight strong repairs, few hints or retries

Starting again

Choose *New mission* from the main menu. The game asks you to confirm, because it replaces your saved mission and all eight repairs. Sound and motion settings are kept. If you are playing in the edu.games player, remember that progress lasts only for the page session.

A CHALLENGE FOR A SECOND RUN

- ☐ Finish all eight repairs with no hints.
- ☐ Hit the move target on at least four repairs.
- ☐ Reach Master engineer.
- ☐ Explain to someone else how you solved the airlock without pressing a single switch first.

Glossary

Algorithm A series of clear steps that solves a problem.

AND A logic rule that is true only when both inputs are true.

Boolean A value that is either true or false (on or off).

Budget A fixed total of a resource that you must plan within.

Capacity The most a shared resource can supply at one time.

Circuit A complete, unbroken path for electricity from a source to a destination.

Constraint A rule or limit that a solution must obey.

Debugging Finding the first wrong step in a program and fixing it.

Dependency A job that must finish before another can begin.

Driver and driven gear The gear turned by the motor, and the gear it turns.

Edge A line joining two nodes in a graph; in the game, a leg of the route.

Failed test A press of the test button that did not restore the system. Costs 75 XP.

Gear ratio The relationship between the number of teeth on two meshed gears.

Graph A set of nodes joined by edges.

Hint One of three escalating clues per repair. Each costs 100 XP.

Interlock A safety rule that prevents a machine operating unless conditions are met.

Inversely proportional When one quantity goes up, the other goes down by the same factor.

Move Any change you make to a puzzle's controls. Counted against the move target.

Move target The number of moves a planned solution needs. Extra moves cost 5 XP each.

New Meridian The colony Aster's garden is travelling to.

Node A point in a graph; in the game, a waypoint.

NOT A logic rule that reverses a condition.

OR A logic rule that is true when at least one input is true.

Parity Whether a count is odd or even.

PIP Aster's small station intelligence and your companion.

Rank Cadet, Technician, Flight engineer or Master engineer, based on total XP.

RPM Revolutions per minute.

Sequence The order in which steps run.

Side effect A change caused somewhere other than where you acted.

Telemetry The live readout of moves, tests and time on a repair screen.

Toggle A switch that flips between two states.

XOR Exclusive or: true when exactly one input is true.

XP Experience points. Each repair earns between 200 and 1000.