Assignment 7: Full Mission: Operation Spotlight

Final briefing, agent: one satellite, three minutes, every trick you've learned, and a rival operative who is learning too.

Your mission

Build your **complete competition program**, then test it against real opponents.

Your program must do all of this at once:

- **Always aim** at the rival satellite.
- **Photograph** them, but only when your guard questions all say yes.
- **Run an item plan** with at least two stops that can never get stuck.
- **Raise the mirror** at the right moment to block their photos.
- Then **scrimmage**: face a classmate's program, lose or win, change ONE thing, and try again.

You'll know it works when: against an opponent that does nothing, your final score is **10 or more** (at least 8 points better than an empty program) and your satellite is always working on something, never drifting around confused.

Read the map the way the official manual draws it: the arena is wide, with **+Y to the right** and **+X up**. The **Light Zone is the left half (−Y)** and the hatched **Dark Zone is the right half (+Y)**, until they swap at **t = 60 s**, and swap again at **t = 150 s**. The **squares** are the asteroids: score items **4, 5, and 6**, worth **+1.5 points** each. The **circles** are energy packs **0–3**; collect one and your energy refills to **5.0**. The **diamonds** are mirrors **7 and 8**, your shield against their camera.

> ⏱ **Every second:** your satellite re-reads your whole main page, top to bottom. It re-aims, re-asks every guard question, checks your step variable, and re-decides about the mirror, 180 times per match. The light zones swap. Items get stolen. Energy drains. Only what your program **re-checks** is true.

Mission rules

1. Create your project on the **AsteroidBee 2026** game, Graphical Editor.
2. Your program must include **automatic aiming**: no photo luck allowed.
3. Your photo block must be protected by a **guard** with at least these questions: is the camera ready? am I facing them? do I have enough energy?
4. Your **item plan** must use a step variable, have **at least two steps**, and must never stall, even if the rival steals your item. (The `no-one` dropdown is your friend here.)
5. You must have a **mirror rule**: a question that decides when the shield goes up.
6. Write a **strategy memo** (5–8 sentences): where your points come from, what your guards check, and what you changed after scrimmaging.
7. Beat the empty-program baseline by **at least 8 points**.

Blocks to explore

- **AsteroidBee MS- Pictures**: your camera, your aim, and the questions that protect them.
- **AsteroidBee MS- Items**: where items are, who holds them right now, and the mirror controls.
- **AsteroidBee MS- Light/Dark**: who is lit, who is hiding, and how long until that flips.
- **AsteroidBee MS- Other**: the live scoreboard and your fuel gauge.
- **SPHERES Controls**: destinations and aiming targets, same as always.
- **Variables**: your plan's only memory between seconds.

- **Logic**: sometimes the most useful question is the opposite of the one printed on the block.
- **Math**: plain numbers for coordinates and energy limits.

Build it, step by step

This build is the whole season on one page: **four standing rules** that the satellite re-reads every second. Build it once exactly as written, watch it win, and then start changing things. Each picture shows exactly what you should see on screen.

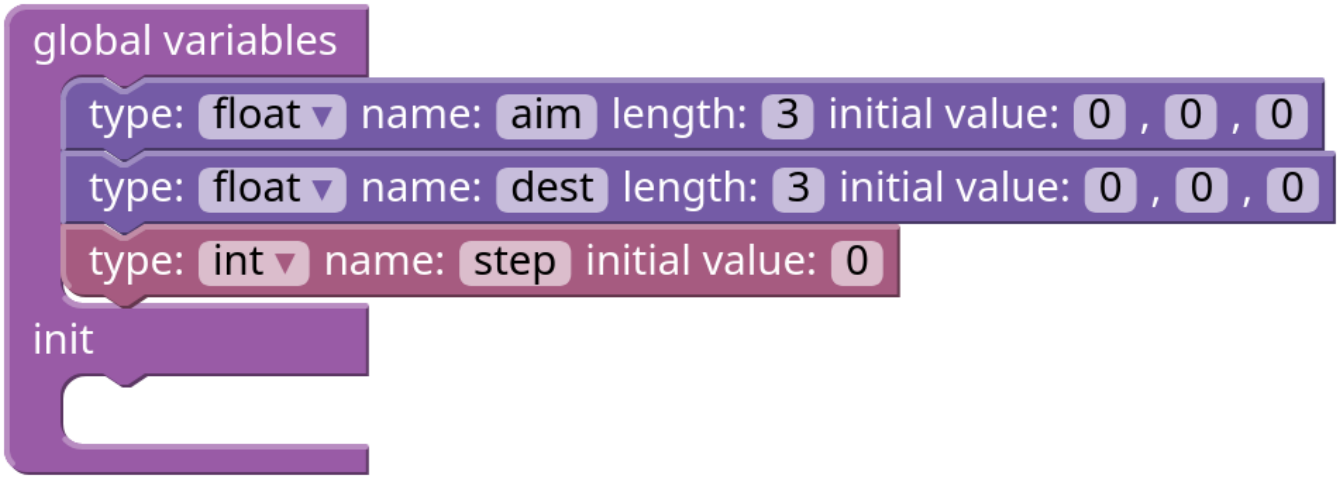
1. Create the project

Make a new project on the **AsteroidBee 2026** game with the **Graphical Editor**. Two pages, as always: the **init** page for your program's memory, the **main** page for the rules.

2. Declare your memory (init page)

On the **init** page, drag **two** `declare array` blocks into the "global variables" slot. Set the first one's type to `float`, name it `aim`, length **3**. Set the second to `float`, name it `dest`, length **3**. `aim` is where the game will write the direction to the rival; `dest` is where it will write item locations.

Snap a plain `declare` block underneath. Switch its type dropdown to `int`, name it `step`, and leave its initial value at **0**. This one number is your plan's entire memory between seconds.



Your init page should match this picture: `aim`, `dest`, `step`. Three memory slots, nothing else.

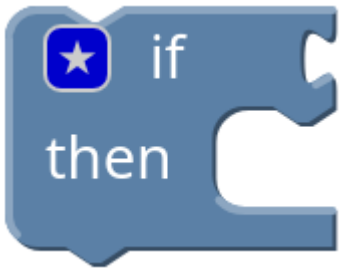
3. Standing rule 1: always aim (main page)

Switch to the **main** page. From **AsteroidBee MS- Pictures**, drag `get att to other to` to the very top of the loop slot. It arrives with a small array block already attached. Open that block's dropdown and pick `aim`. Every second, the game writes the direction to the rival into `aim`.

From **SPHERES Controls**, snap `set PositionTarget ▾` directly below it, then switch its dropdown to **`set AttitudeTarget`**. From **Variables**, plug the whole-array `aim` block into its socket. Same palette block, different job: it now turns your satellite to face wherever `aim` points.

These two blocks run first, every second, no questions asked. That is what "always aim" means: re-aimed 180 times per match.

4. Standing rule 2: photograph, behind five questions



From **Logic**, snap an `if / then` block below the aiming pair. This is your photo guard.

You need room for **five** questions. Drag an `and` block into the `if / then` block's question socket, then plug a second `and` into the first one's left opening, a third into that one, and a fourth into that: **four `and` blocks, five empty openings**. Leave every dropdown on `and`: all five questions must say yes.

Now fill the five openings, in this order:

Question 1: `is camera on` (from **AsteroidBee MS- Pictures**), false during the camera's 3-second cooldown.

Question 2: `is facing other`: am I actually pointed at them yet?

Question 3: `check in light` (from **AsteroidBee MS- Light/Dark**); switch its dropdown to **`other`**. No photographing a rival who's hiding in the dark.

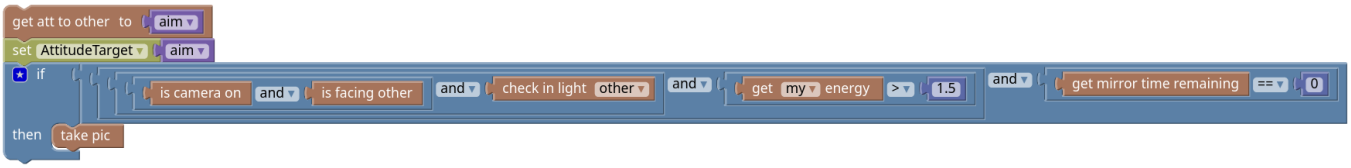
Question 4 is built from a **Logic** compare block. Its dropdown offers `==`, `!=`, `<`, `<=`, `>`, `>=`. Set it to **`>`**.

Plug `get my energy` (from **AsteroidBee MS- Light/Dark**) into its left side...

...and a **Math** number block reading **1.5** into its right side. The question now reads: `get my energy` **`>`** `1.5`.

Question 5: another compare block, set to **`==`**. Left side: `get mirror time remaining` (from **AsteroidBee MS- Items**). It reads the seconds left on your active mirror, and **0** means none is running. Right side: a number block reading **0**.

Finally, snap `take pic` into the `then` slot. It only ever fires when all five answers are yes.



Compare against this picture: the aiming pair on top, then one `if / then` holding five questions and one `take pic`.

5. Standing rule 3, first stop: the diamond

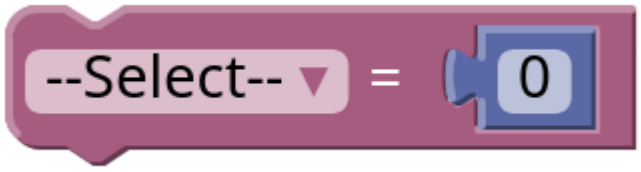
Your item plan is the state machine from Assignment 6, with one upgrade: it advances even when the rival steals your prize.

Snap a new `if / then` below the photo guard. Into its question socket put a compare block set to `==`, with the `step` variable block (from **Variables**; it shows just the name) on the left and a number block reading **0** on the right: `step == 0`.

Inside the `then` slot, in order: first `get item … loc to` (from **AsteroidBee MS- Items**). Set its item dropdown to **7** and its attached array block's dropdown to `dest`. Item 7 is one of the **diamonds**: a mirror, and your shield rule below will need it.

Next, another `set PositionTarget ▾` block (this time **leave the dropdown on `set PositionTarget`**) with the whole-array `dest` block plugged in. Destination: wherever item 7 is, re-checked every second.

Still inside the `then` slot, add an inner `if / then`. Plug a `not` block into its question, and `check have item` into the `not`: item dropdown **7**, holder dropdown **`no-one`**. Read it out loud: "if it's NOT true that no-one has item 7", meaning somebody picked it up. You, or the rival. Either way, this stop is over.



Inside that inner `then` slot: the variable set block, which reads `step =`. Pick `step` and plug in a number block reading **1**. That is the no-stall trick: the plan advances the moment the item is gone, not only when it's yours.

Your first branch should match this picture exactly.

6. Second stop: the square

Build the same branch again, right below; only the numbers change. Outer question: `step == 1`. Inside: `get item … loc to` set to item **5** filling `dest`, then `set PositionTarget` with `dest`, then the inner `if / then` with `not` + `check have item` **5** `no-one`, setting `step =` **2**. Item 5 is one of the **squares**: an asteroid, worth **+1.5 points**.

7. Third stop: park

One more `if / then`, question `step == 2`. Inside its `then` slot, a single `setPos` block (from **SPHERES Controls**) with **0.25**, **−0.2**, **0** typed into its three slots. With the errands done, your satellite parks there, and the photo guard above keeps shooting for the rest of the match.

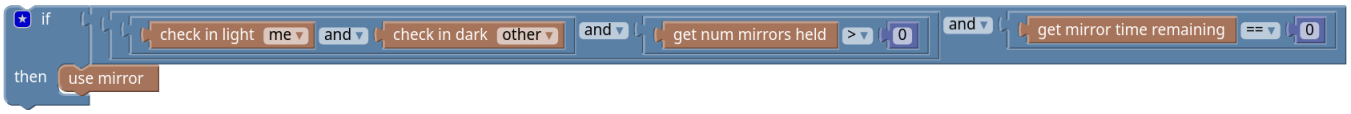
8. Standing rule 4: the mirror

Last rule, at the bottom of the page: one `if / then` with **four** questions (three `and` blocks this time), then the shield.

Question 1: `check in light`, dropdown left on `me`: I'm exposed. **Question 2:** `check in dark`, dropdown switched to `other`: they're hiding.

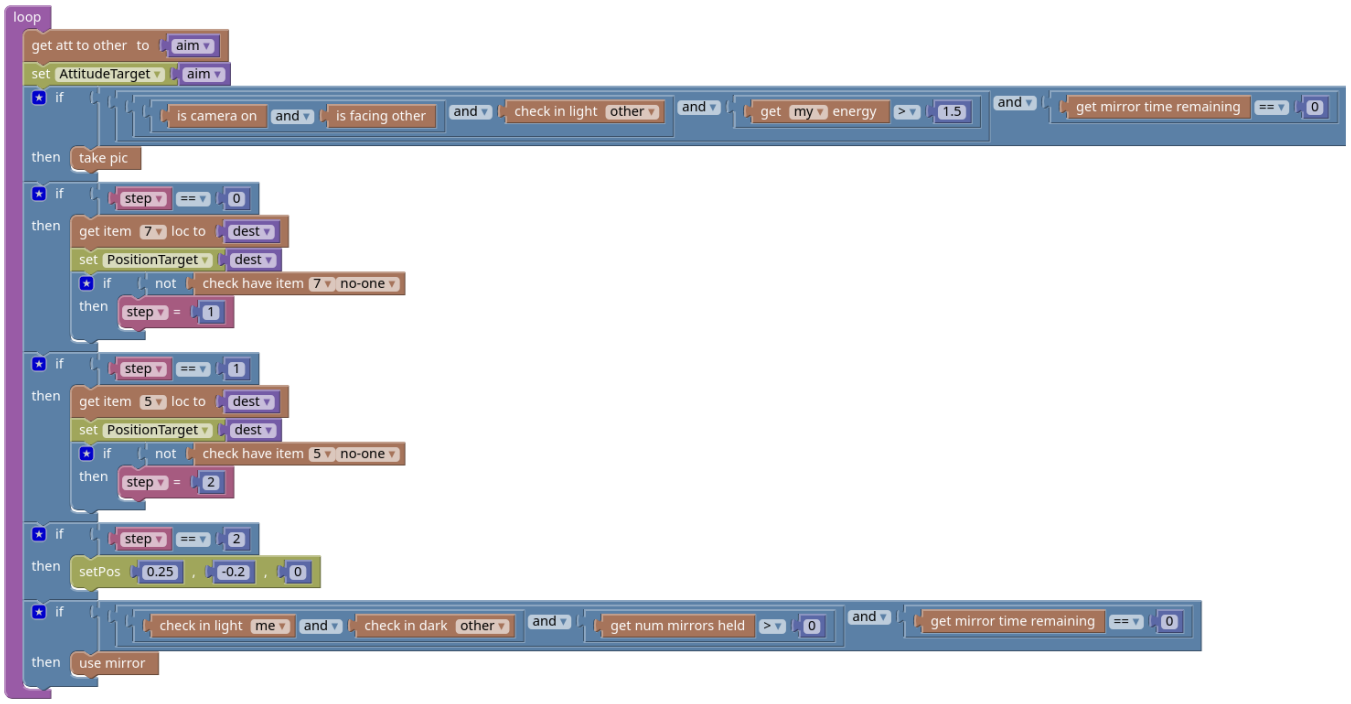
Question 3: a compare block set to `>`: `get num mirrors held` on the left, a number block reading **0** on the right, meaning I actually have a mirror (you collected the diamond in step 0). **Question 4:** a compare set to `==`: `get mirror time remaining` and **0**, meaning no mirror already running.

In the `then` slot: `use mirror`. The shield lasts 24 seconds, and this rule re-asks its four questions every second after it ends.



Compare against this picture before the final check.

9. Compare your whole workspace



Your main page, top to bottom, should match this: the aiming pair, the five-question photo guard, the three step branches (`step == 0`, `1`, `2`), and the mirror rule. Four standing rules, all re-read every second.

10. Simulate, then scrimmage

Simulate the full **180 seconds as Blue** against an idle opponent and walk through "Check your work" below. Then the real test: face a classmate's program. Win or lose, change **ONE thing** (one guard question, one destination, one threshold) and run it again. Write your **strategy memo** (5–8 sentences) while the second match is fresh.

Hint

Champions do not write longer plans; they write better **questions**. Every guard in your `if / then` blocks is a question the satellite re-asks every second. Make three lists: the questions that should **stop a photo**, the questions that should **advance your plan**, and the one question that should **raise the mirror**.

Check your work

Watch the simulation viewer and the log:

- Against an idle opponent: your score climbs steadily and finishes at **10 or more**. No stretch of more than 30 seconds where nothing is attempted.
- Early in the match, your satellite picks up an item that's part of your plan, and if the rival steals it first, your satellite moves on to the next step instead of parking on an empty spot.
- When the rival is hiding in the dark, your camera **holds its fire**. Check the log: no failed pictures wasting 1.0 energy each.
- When YOU are in the light and they're hiding, and you hold a mirror that isn't already running: the mirror goes up (it lasts 24 seconds).
- At t = 60, the light and dark halves swap. Does your program notice, or does it keep acting like it's still second one?
- Your strategy memo is written: 5–8 sentences.

Reflection (write 3–5 sentences)

Explain your block program in your own words, as if to a teammate who missed class: **what does your program do every second?** Then answer: which questions stop a photo, which questions move your plan forward, and which one question raises the mirror?