Assignment 7: Full Mission: Operation Spotlight

Name: _____ Date: _____

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 arena map (on the projector and the handout) 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. The pictures are on the projector and the handout.

☐ 1. **Create the project** on the **AsteroidBee 2026** game with the **Graphical Editor**. Two pages, as always: **init** for memory, **main** for the rules.

☐ 2. **Declare your memory (init page).** Drag **two** `declare array` blocks into the "global variables" slot: `float` `aim`, length **3**, and `float` `dest`, length **3**. Underneath, a plain `declare` block: type `int`, name `step`, initial value **0**. Three memory slots, nothing else.

☐ 3. **Standing rule 1: always aim (main page).** At the very top of the loop slot: `get att to other to`, with its attached array block's dropdown set to `aim`. Directly below: `set PositionTarget ▾` with its dropdown switched to **`set AttitudeTarget`** and the whole-array `aim` block plugged in. These two run first, every second: re-aimed 180 times per match.

☐ 4. **Standing rule 2: photograph, behind five questions.** Snap an `if / then` below the aiming pair. Chain **four `and` blocks** in its question socket to make five openings, then fill them in order: `is camera on`; `is facing other`; `check in light` with its dropdown switched to **`other`**; a compare block set to **`>`** reading `get my energy` `>` `1.5`; a compare block set to **`==`** reading `get mirror time remaining` `==` `0` (it reads the seconds left on your active mirror; 0 means none is running). In the `then` slot: `take pic`. It only ever fires when all five answers are yes.

☐ 5. **Standing rule 3, first stop: the diamond.** A new `if / then` with a compare set to **`==`**: the `step` variable block (it shows just the name) `==` `0`. Inside, in order: `get item … loc to` with item dropdown **7** filling `dest`; `set PositionTarget` (dropdown left alone this time) with `dest` plugged in; then an inner `if / then` whose question is `not` wrapped around `check have item`: item **7**, holder **`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. Inside the inner `then`: the variable set block reading `step =` with **1**. That is the no-stall trick: the plan advances the moment the item is gone, not only when it's yours. Item 7 is one of the **diamonds**: a mirror, and rule 4 below will need it.

☐ 6. **Second stop: the square.** Build the same branch again, right below; only the numbers change. Outer question `step` `==` `1`; inside, item **5** everywhere item 7 was, and `step =` **2** at the end. 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: a single `setPos` block with **0.25**, **−0.2**, **0** typed into its three slots. Errands done, your satellite parks there, and the photo guard above keeps shooting.

☐ 8. **Standing rule 4: the mirror.** At the bottom of the page: one `if / then` with **four** questions (three `and` blocks): `check in light` left on **`me`** (I'm exposed); `check in dark` switched to **`other`** (they're hiding); a compare set to **`>`** reading `get num mirrors held` `>` `0` (I actually have a mirror); a compare set to **`==`** reading `get mirror time remaining` `==` `0` (none 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.

☐ 9. **Compare your whole workspace** against the finished-layout picture on the projector: 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 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?