Assignment 7: Full Mission: Operation Spotlight — Teacher Guide

At a glance

Learning objective	Capstone: students combine an always-on aiming + guarded camera (Assignment 5), an item plan on the state machine (Assignment 6), a mirror defense rule, and energy/fuel discipline into one complete competitive strategy, then scrimmage against classmates, iterate, and turn in a written strategy memo.
Time estimate	2–3 class sessions
Project setup	New project on the **AsteroidBee 2026** game, using the **Graphical Editor**
Prerequisite assignment	Assignments 1–6
Blocks introduced	`use mirror`, `get num mirrors held`, `get mirror time remaining`, `check in dark other`, `check in light me`, `check have item _ no-one`, `not` (Logic), `get my score` / `get other score`, `get remaining fuel`

The one idea this assignment teaches

A competitive strategy is not a longer to-do list; it is a better set of **standing questions**. Everything the class has built so far now lives together on one main page: the aiming pair, the guarded camera, the step-by-step item plan, and (new this week) a mirror rule. The main page re-runs every second, so each of those rules gets re-asked every second, all match long. The skill students are practicing is making several independent rules share one page without fighting each other: each rule is its own `if … then …` block with its own questions, and the satellite's whole "personality" is just which questions it asks before acting.

The second half of the idea: **nothing in this game stays the way the briefing said.** The light half becomes the dark half at t = 60 and again at t = 150. Items get stolen. Energy drains. The opponent moves. A program that checked a fact once at the start of the match (or worse, never checked and just assumed it) will quietly go wrong halfway through. The cure is the habit students have been building all season: every condition that matters gets a question block inside an `if`, and that question gets re-asked every second. The new `not` block earns its keep here, because sometimes the available question (`check have item 7 no-one`, "does nobody hold item 7?") is the opposite of the question you actually want to act on ("has item 7 been claimed by anyone?").

> ⏱ **Every second:** the satellite re-reads the whole main page from top to bottom: it re-aims at the opponent, re-asks all five photo-guard questions, re-checks which step the plan is on, and re-asks the mirror questions. The program never "moves past" any of these rules; they are all live, every second, for all 180 seconds. Champions don't write longer plans; they write better questions.

This is also the assignment where students learn that strategies are compared, not just completed. They will scrimmage against classmates' programs, lose, change exactly one guard or one step, and write down why. The strategy memo is where that thinking becomes visible. Collect it like any other deliverable.

Before class

- [ ] Re-read the game rules, especially the five photograph conditions (note: a photo fails if **either** satellite has an active mirror), the item table, the light/dark swaps at t = 60 and t = 150, and the 60-second thruster-fuel budget.
- [ ] Build the reference solution yourself from reference-solution.md and simulate it against an idle opponent (an opponent project with an empty program), so you know what healthy behavior looks like.
- [ ] Run an **empty** graphical project on the same game once and write down its final score. That number is the "empty-program baseline"; students must beat it by at least 8 points.
- [ ] Decide your scrimmage logistics: in the simulation settings you can choose a classmate's submitted project as the opponent. Decide whether teams pair up and run both matchups, or you keep a sign-up board.
- [ ] Print the printable packet, one per student. It includes the strategy-memo requirement.
- [ ] Clear whiteboard space for the points-budget discussion (step 1 below).

Walkthrough: what to say and do

1. **Start at the whiteboard, not the computer: the points budget.** Write two facts: a successful photo is worth about 2.3–3.0 points and you can take one over and over (the camera only needs 3 seconds between shots); a score item is worth 1.5 points, once. Ask each team: **"Where will YOUR roughly 20 points come from?"** Make them write a budget, e.g. "6 photos + 1 score item ≈ 16–19." Teams that skip this step build five-stop scavenger hunts (see Mistakes, below). Their budget becomes the first sentence of their strategy memo.

2. **Walk through the anatomy of the reference strategy on the projector.** Say this out loud, and mean it: this is ONE good strategy, not THE answer. A team that copies it will lose to a team that iterates on its own idea. Then narrate what the program does every second: (a) ask the game where the opponent is, (b) aim at them (aiming is essentially free, so we do it unconditionally), (c) shoot only if every guard question says yes, (d) run the item plan for whichever step we're on, (e) raise the mirror if the defense questions say yes. Five rules, one page, all re-read every second.

3. **Build the always-on aiming pair (review from Assignment 5).** On the init page, in the "global variables" slot: an array declare for `float aim` with length 3, another for `dest`, and a plain declare for `int step` with initial value 0. On the main page, drag `get att to other to` from AsteroidBee MS- Pictures. Its array block comes pre-attached; pick `aim` from the dropdown. Below it, drag `set PositionTarget ▾` from SPHERES Controls and **switch its dropdown to `set AttitudeTarget`**, with the whole-array `aim` block in its socket. These two sit at the very top, outside any `if`.

4. **Build the photo guard.** One `if … then …` with `and` blocks chaining five questions: `is camera on`, `is facing other`, `check in light other`, a compare block reading `get my energy` **>** `1.5`, and a compare block reading `get mirror time remaining` **=** `0`. Inside the `do`: `take pic`. Pause on that last condition and ask the class: **"Why is MY mirror in MY camera's guard?"** Answer: a photo fails if either satellite has an active mirror; an active mirror blocks your own camera too. Shooting through your own mirror pays 1.0 energy for 0 points, every 3 seconds. (The 1.5 energy floor keeps a reserve so one failed shot can't zero them out.)

5. **Build the item plan on the state machine** ("state machine" is just the pattern from Assignments 4 and 6: a `step` variable that remembers where we are, plus `if` blocks that act on it). Three `if` blocks in a row:

6. `if step = 0`: `get item 7 loc to` filling `dest`, then `set PositionTarget` with `dest`. Why item 7? Look at the arena diagram: it's a **mirror** item, it starts in our light half, and it's close to Blue's start, a cheap first stop that buys our defense.
7. Nested inside step 0: `if not ( check have item 7 no-one )` → `set step = 1`.
8. `if step = 1`: same shape, item 5 (a score item), advancing to step 2 the same way.

9. `if step = 2`: `setPos 0.25, -0.2, 0`: park there. There is no `take pic` in step 2: the photo guard at the top of the page is already firing whenever its questions say yes. The plan and the camera are separate rules sharing the page.

10. **Explain the `no-one` dropdown (say this carefully, it's the anti-stall trick).** `check have item 7 no-one` asks "does nobody hold item 7?" Wrapped in `not`, it becomes "has item 7 stopped being unclaimed?", which is true whether **we** grabbed it or the **opponent** stole it. Either way, the plan moves on. Ask the class: **"What happens if we advance on `check have item 7 me` instead, and Red gets there first?"** (Step 0 forever: a satellite parked on an empty spot for the rest of the match.) The plan must advance on what the game confirms, not on what we hoped.

11. **Build the mirror rule.** One more `if` with four questions chained by `and`: `check in light me`, `check in dark other`, `get num mirrors held > 0`, and `get mirror time remaining = 0`. Inside: `use mirror`. Translate it for the class: "I can be photographed right now, my own camera is useless anyway because they're hiding in the dark, I'm carrying a mirror, and one isn't already running: shields up." The 24-second shield costs us nothing we could have used (we couldn't shoot a hidden opponent anyway).

12. **Simulate against an idle opponent.** Healthy behavior: grabs the mirror item early, photographs steadily whenever the opponent is photographable, collects item 5, parks at (0.25, −0.2) and keeps shooting. The score should climb past ~15 by match end and there should be no long dead stretches.

13. **SCRIMMAGE.** In the simulation settings, choose a classmate's submitted project as the opponent (or pair teams so both matchups get run). Expect noise and gloating. Channel it into step 10.

14. **Iterate with discipline.** Rule for the room: **each scrimmage loss changes exactly ONE guard or ONE step**, and the change gets a line in the strategy memo: what changed, why, what happened in the rematch. One change at a time is how they learn which question mattered.

15. **The fuel/time reality check.** Each satellite has **60 seconds of total thruster firing time** (`get remaining fuel` reads what's left). Crossing the arena costs several seconds of it, and so does the travel time. Three items scattered across the map is usually too greedy. Ask each team: "How many arena crossings does your plan ask for? What would you score during those seconds if you stayed put and shot instead?"

16. **Collect the strategy memo** (5–8 sentences: where the points come from, what the guards are, what changed after scrimmaging). The memo is required by the mission rules.

The mistakes you will see

1. Mirror friendly-fire

- **What it looks like:** mirror up, score frozen, energy draining: the satellite keeps shooting doomed photos through its own shield, 1.0 energy each, for 0 points.
- **Why it happens:** students file the mirror under "defense only" and never connect it to their own camera. The photo rules say a picture fails if either satellite has an active mirror, including yours.
- **The question to ask:** "Read the photo rules again: whose mirrors matter?" (Both! That's why the reference guard includes `get mirror time remaining` = 0 on the **camera**, not just on the mirror rule.)

2. Greed: the five-stop item tour

- **What it looks like:** the satellite spends the whole match commuting between items, burns through its 60 seconds of fuel, and takes almost no photos. Final score: a handful of 1.5s and nothing else.
- **Why it happens:** items feel like guaranteed points, and travel feels free. Students underestimate how long crossings take and forget that photos are repeatable.
- **The question to ask:** "What does one photo earn vs one score item, and which can you do twenty times?"

3. Plans that ignore the clock (the "checked once, true forever" trap)

- **What it looks like:** the satellite camps the y < 0 half all match "because it is the Light Zone". Then at t = 60 that half becomes the dark, energy quietly stops recharging, and the score stalls.

- **Why it happens:** this is the same root confusion as "I expected my blocks to run in sequence over time," in a new costume. Students treat a fact from the t = 0 briefing as if the program had checked it and could move on. But the main page doesn't move on: it re-runs every second, and only what it re-checks is true. Zones swap, items vanish, energy drains; a baked-in assumption is a question the program never asks.
- **The question to ask:** "What does `get light switch time` read right now? What changes when it hits zero?" Then: "Which of your blocks would notice?", building the per-cycle habit that conditions must be RE-CHECKED, not assumed from the briefing.

Reference solution at a glance

The **init page** declares two 3-slot `float` arrays, `aim` and `dest`, and an `int` named `step` with initial value 0. The **main page**, top to bottom: the aiming pair (`get att to other to` into `aim`, then `set AttitudeTarget` with `aim`); the five-question photo guard wrapped around `take pic`; three step `if`s, each advancing `step` with `not` around a `check have item _ no-one` question: step 0 flies to item 7 (mirror), step 1 to item 5 (score), step 2 parks at (0.25, −0.2); and finally the four-question mirror rule wrapped around `use mirror`. The full block layout and the exact generated C are in reference-solution.md. Remind students (and yourself): entirely different strategies are equally valid. What counts is coherent guards and a non-stalling plan, not similarity to this one.

Wrap-up questions

1. Your photo guard is a list of questions. Which single question saved you the most energy or points in scrimmages, and what did you see in the simulation that proves it?
2. At t = 60 the light and dark halves swap. Walk me through the very next second of your program: which question blocks notice the change, and what does the satellite do differently because of them?
3. Two teams can submit completely different strategies and both earn full marks. What makes a strategy "coherent"? What ONE change after a scrimmage made the biggest difference for your team, and how do you know?