Assignment 3: Am I There Yet?

Name: ____ Date: __

Mission tagline: a spy who doesn't know where they are isn't much of a spy. Teach your satellite to notice when it reaches the drop point, and say so.

Your mission

Headquarters has a drop point for you: **(0, −0.4, 0)**. Fly there from Blue's starting spot at (0.4, −0.6). You already know the block for that from last time.

But this mission has a twist. Your satellite must **sense its own position** every second and **announce when it has arrived**. Success looks like this: the log stays quiet while you fly… and then **"Arrived!"** appears when you reach the drop point.

To pull this off, your satellite needs a notebook: a 12-slot list (an array) where it writes down its position every second. **Slot 0 is your X. Slot 1 is your Y.** Heads up: the block that creates the notebook lives on the **init page**, in the "global variables" slot. You won't find it while you're on the main page.

> ⏱ **Every second:** the satellite re-reads your whole main page from the top. It senses its position again, sets its destination again, and asks your `if` question again, fresh, every single second, even after it has already arrived. Watch what that does to your message.

Mission rules

- Your project must be created on the game **AsteroidBee 2026** (Graphical Editor).
- The target is **(0, −0.4, 0)**.
- "Arrived" means: within **0.05 m (5 cm)** of the target on X **and** on Y.
- Announce arrival with a **DEBUG** message in the log.

Blocks to explore

- **Variables**: your satellite's memory. The block that creates an array only shows up here while you're on the **init page**.
- **SPHERES Controls**: one block in here can fill your whole array with the satellite's position, speed, and facing. You already know the block that flies you somewhere.
- **Logic**: for asking a yes/no question and doing something only when the answer is yes. Two questions can be glued into one.
- **Math**: numbers, plus a block with a dropdown hiding exactly the math a "how far away am I?" question needs.
- **Debug**: for making your satellite talk.

Build it, step by step

Follow these steps in order. The block pictures are on the projector and on the handout.

☐ **1. Create the project.** Create a new **Graphical Editor** project on the game **AsteroidBee 2026**. You start on the **main** page, whose root block is the `loop`. Everything you snap into it runs **once per second, every second, for the whole match**.

☐ **2. Make the notebook (on the init page).** Switch to the **init** page. From **Variables**, drag the array declare block into the **global variables** slot. Leave the type on **float**, name it **myState**, set the length to **12**, and leave all twelve `initial value` slots at 0.

☐ **3. Sense your position (back on the main page).** From **SPHERES Controls**, drag out `get My ZRState`. It arrives with a whole-array block already in its socket. Set that dropdown to **myState**. Snap it into the top of the loop slot. Now myState[0] is your X and myState[1] is your Y, fresh every second.

☐ **4. Fly to the drop point.** From **SPHERES Controls**, snap `setPos` underneath `get My ZRState` and type **0**, **−0.4**, **0** into its sockets. Re-issuing the destination every second is exactly right.

☐ **5. Add the question.** From **Logic**, snap the `if / then` block underneath `setPos`. Its empty diamond socket is where your "am I there yet?" question goes.

☐ **6. Build the "am I there yet?" question.** You're building: is my X within 0.05 of 0, and is my Y within 0.05 of −0.4? Assemble it in an empty patch of the workspace: - From **Logic**: one `and` block, plus **two** compare blocks with their dropdowns set to **<** (the choices are ==, !=, <, <=, >, >=). Plug one compare into each side of the `and`. - From **Math**: two of the single-input block (it comes out reading `square root`). Switch each dropdown to **absolute value**. Plug one into the **left** socket of each compare. - From **Variables**: two array slot get blocks (`myState [ index ]`), both set to **myState**. - **X half:** `myState` index **0** goes straight inside the first `absolute value`. - **Y half:** from **Math**, an arithmetic block left on **+**, with `myState` index **1** on its left and a number block reading **0.4** on its right. That whole `+` goes inside the second `absolute value`. - From **Math**: two number blocks reading **0.05**, one in the **right** socket of each compare. - Plug the finished question into the diamond socket of your `if / then`.

☐ **7. Say so.** From **Debug**, snap the `DEBUG` block **inside** the `if / then` block's "then" slot, not below the `if`! Drag a text block into its socket and type **Arrived!** between the quotes.

☐ **8. Compare against the finished layout** (on the projector/handout). Main page, top to bottom: `get My ZRState`, `setPos`, `if / then` with `DEBUG` inside (sense, fly, ask, announce). Init page: the **myState** array declare. If both match, you're ready to fly.

Hint

The satellite can read its own position into a 12-slot notebook every second: slot 0 is X, slot 1 is Y. "Close enough" is a math question: how far am I from the target, and is that distance smaller than 0.05?

Check your work

Run the simulation for about 90 seconds as Blue. You should see:

1. The satellite leaves (0.4, −0.6) and settles at (0, −0.4) in the viewer.
2. The log is **quiet** during the whole flight.
3. Then **"Arrived!"** appears, and keeps appearing, **once every second**, until the end of the run.

If the message never shows up even though the satellite is clearly parked at the target: watch pX and pY in the viewer. How close does it really get? Check which array slot you're testing, check your signs (the target Y is negative 0.4), and make sure your "close enough" bubble isn't impossibly small.

If the message printing over and over surprises you — good. That's the question in your reflection.

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 this: **why does "Arrived!" print every second after you get there, instead of just once?**