Assignment 3: Am I There Yet?

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. After each step, your workspace should look like the picture.

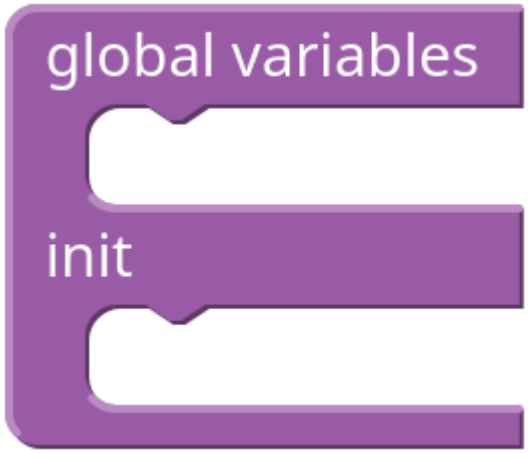
Step 1: Create the project

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 this slot runs **once per second, every second, for the whole match**.

Step 2: Make the notebook (on the init page)

Switch to the **init** page. Its root block has the "global variables" slot:

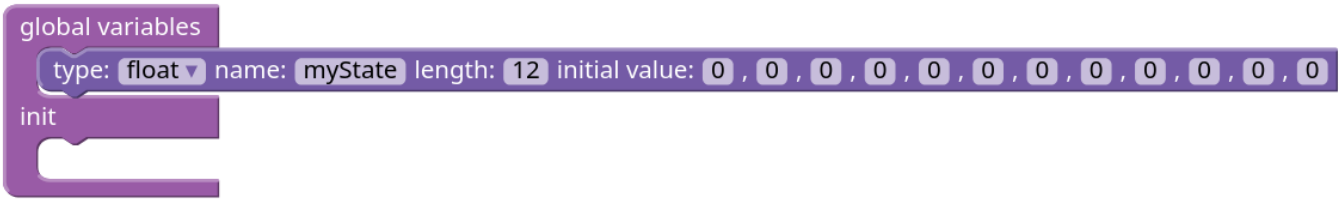


Open the **Variables** category and drag out the array declare block:

Snap it into the **global variables** slot. Then set it up:

- Leave the type dropdown on **float**.
- Click the name and type **myState**.
- Set the length to **12**. (Twelve `initial value` slots appear. Leave them all at 0.)

Your init page should now match this:



Step 3: Sense your position (back on the main page)

Switch back to the **main** page. Open **SPHERES Controls** and drag out the `get My ZRState` block:

It arrives with a whole-array block already plugged into its socket. Click that array's dropdown and pick **myState**. Snap the whole thing into the top of the loop slot. Now, every second, the satellite writes its position into your notebook: **myState[0] is X, myState[1] is Y**.

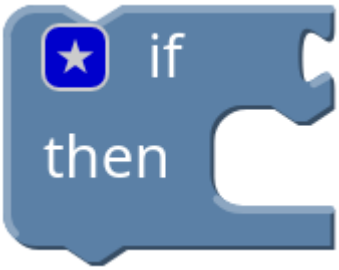
Step 4: Fly to the drop point

From **SPHERES Controls**, drag out the `setPos` block:

Snap it **underneath** `get My ZRState`. Type the drop point into its three number sockets: **0**, **−0.4**, **0**. Re-issuing this destination every second is exactly right: it's a destination, not a push.

Step 5: Add the question

From **Logic**, drag out the `if / then` block:



Snap it underneath `setPos`. Its empty diamond socket is where your "am I there yet?" question will go. You'll build that question next, then plug it in.

Step 6: Build the "am I there yet?" question

This is the big one. You're building: is my X within 0.05 of 0, and is my Y within 0.05 of −0.4? Work on it in an empty patch of the workspace, then plug it in.

First, from **Logic**, drag out the `and` block, which glues two yes/no questions into one:

Next, drag out **two** compare blocks from **Logic** (one for X, one for Y). Set each one's dropdown to **<** (the choices are ==, != , <, <=, >, >=):

Plug one compare block into each side of the `and`.

Now the math. From **Math**, drag out two of the single-input block (it comes out reading `square root`):

Click each one's dropdown and switch it to **absolute value**. That's the "how far away am I, ignoring direction" math:

Plug one `absolute value` block into the **left** socket of each compare block.

From **Variables**, drag out the array slot get block, which reads one slot of your notebook, like `myState[0]`:

You'll need two of them. Set both name dropdowns to **myState**.

- **The X half:** put `myState` with index **0** straight inside the first `absolute value`. (The target X is 0, so your distance from it is just |myState[0]|.)
- **The Y half:** the target Y is −0.4, so the distance is |myState[1] + 0.4|. From **Math**, drag out an arithmetic block, leave its dropdown on **+**, put `myState` with index **1** on its left and a number block reading **0.4** on its right. Then put that whole `+` inside the second `absolute value`:

Finally, from **Math**, drag out two number blocks and type **0.05** into each:

Plug one into the **right** socket of each compare block. The assembled question should look exactly like this:

Plug the whole thing into the diamond socket of your `if / then` block.

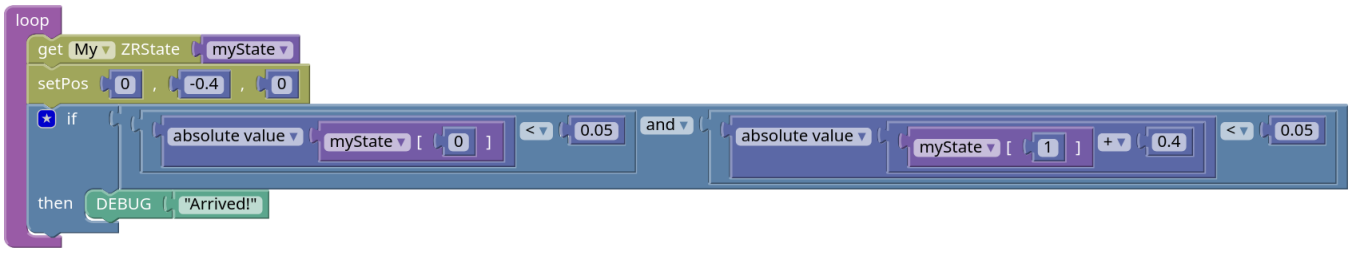
Step 7: Say so

From **Debug**, drag out the `DEBUG` block:

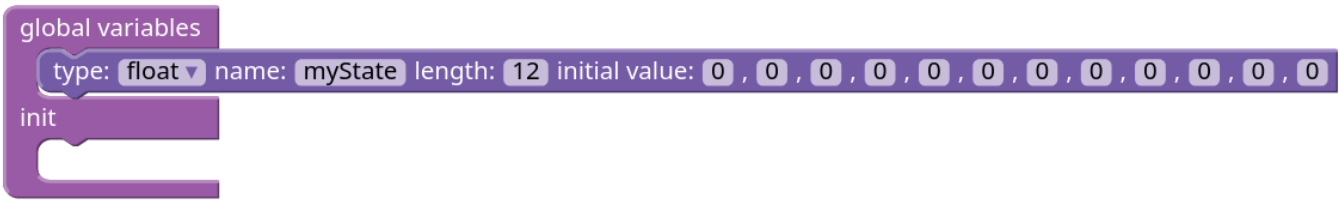
Snap it **inside** the `if / then` block's "then" slot, not below the `if` ! It should only run when the answer is yes. Then, also from **Debug**, drag a text block into the DEBUG block's socket and type **Arrived!** between the quotes:

Step 8: Compare against the finished layout

Your main page should now match this, top to bottom (sense, fly, ask, announce):



And your init page should still match Step 2:



If both pages 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?**