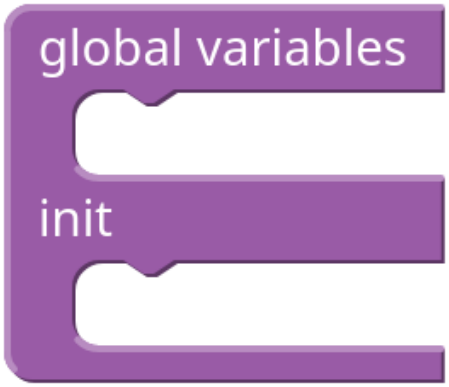
Block Dictionary — every block, pictured, with the C it generates

Every block named in these assignments, shown exactly as it appears in the graphical editor (each picture below is rendered through the production Blockly bundle, so it matches the editor pixel-for-pixel), with what it does and the C code it generates (visible with the IDE's "C Code" toggle).

You can tell what a block is by its connectors: a block with a notch on top and a bump underneath snaps into a stack: it DOES something. A block with a puzzle tab on its left edge plugs into a socket: it REPORTS a value.

The two pages every project has

Page	Root block	Generates	Runs
init		`void init()` `{ … }`	**once**, at the very start of the match
main		`void loop()` `{ … }`	**once per second, every second, for the whole match**

These root blocks can't be deleted or moved. Variable-**declare** blocks only fit in the "global variables" slot (and only appear in the palette while you're on the init page). Movement and game blocks only work on the main page (or in a function page).

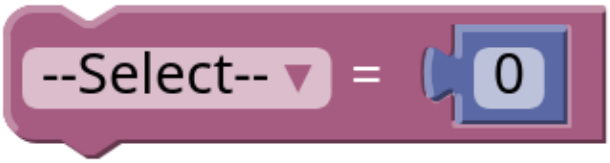
SPHERES Controls (movement & senses, available in every project, including Free Mode)

Block	What it does	Generated C
	Fly to the point (X, Y, Z) and stop there. Re-issuing it every cycle is correct: it's a destination, not a push.	`setPos(x, y, z);` (a helper that calls `api.setPositionTarget(pos);`)
	Dropdown also offers `set AttitudeTarget` (point at a direction vector), `set VelocityTarget`, `set AttRateTarget`, `set Forces`, `set Torques`. Takes a whole 3-element array variable.	`api.setAttitudeTarget(arr);` etc.
	Fills a 12-slot array with my state. Dropdown also offers `Other` (the opponent!). Slots: **[0]=x, [1]=y, [2]=z**, [3..5]=velocity, [6..8]=facing direction, [9..11]=spin rates.	`api.getMyZRState(arr);`
	Seconds since the match started (0 on the first loop).	`api.getTime()`
	True if I'm that satellite (there is a matching `I am Red`).	`(api.getSatColor()==SPHERE1)`

Debug

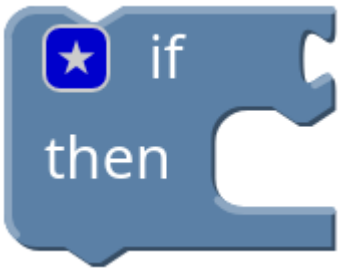
Block	What it does	Generated C
	Prints a message to the simulation log.	`DEBUG(( "message" ));`
	The message. Type your words **between the quotes**.	`"message"`

Variables

Block	Where	Generated C
	**init page, "global variables" slot only**	`float name;` at the top of the file, plus `name = 0;` inside `init()`
 (typing a bigger length grows one `initial value` slot per element)	same	`float name[12];` plus per-slot initial values in `init()`
 variable **get** (the block shows just the name)	anywhere	`name`
 variable **set**	anywhere	`name = value;`
 array slot get	anywhere	`name[index]`
 array (whole)	plugs into array sockets like `get My ZRState`, `get item loc to`, `set AttitudeTarget`	`name`

> The "whole array" block comes **pre-attached** when you drag `get My ZRState`, `get att to other to`, or `get item … loc to` from the palette; students pick their array's name from its dropdown.

Logic

Block	Notes	Generated C
	The gear icon adds `else if` / `else` arms.	`if (cond) { … } else { … }`
	Dropdown: `==`, `!=`, `<`, `<=`, `>`, `>=`.	`a == b`, `a < b`, …
	Dropdown: `and` / `or`.	`a && b` / `a || b`
	Flips true/false.	`!a`
	Dropdown: `true` / `false`.	`true`

Math

Block	Notes	Generated C
	Type any number, e.g. `0.5` or `-0.4`.	`0`
	Two-number arithmetic; dropdown: `+ − × ÷ ^`.	`a + b`, …
	The dropdown includes **`absolute value`**, the form used in Assignments 3–4: 	`sqrtf(a)`, **`fabsf(a)`**
	Adds to a variable in place.	`name += v;`

Loops: ⚠ a careful note

`repeat 10 times`, `while`, and `for` blocks repeat **within a single one-second cycle**. They do NOT make something happen "for 10 seconds." Anything that should unfold over match time must be done with a variable + `if` across cycles (the state-machine pattern, Assignment 4). These assignments deliberately never need a loop block.

AsteroidBee category (only in projects created on the AsteroidBee 2026 game)

AsteroidBee MS- Pictures

Block	What it does	Generated C
	Attempt a photograph (costs 1.0 energy, 3 s cooldown, even on failure).	`game.takePic();`
	Same attempt, but **reports the points earned** (0 if it failed) so you can store or compare it.	`game.takePic()`
	Fills a 3-slot array with the direction that points at the opponent. Feed it to `set AttitudeTarget`.	`game.getAttToOther(arr);`
	Am I aimed at the opponent (within the camera cone)?	`game.isFacingOther()`
	What a picture would score right now (0 = it would fail). Costs 0.1 energy.	`game.getPicPoints()`
	False during the 3 s cooldown.	`game.isCameraOn()`

AsteroidBee MS- Items

Block	What it does	Generated C
	Who holds item N? The second dropdown asks about `me`, `other`, or `no-one`: three questions in one block.	`game.checkHaveItem(4)` / `…Other(4)` / `…NoOne(4)`
	Activate a held mirror (24 s shield). (A socket version also exists that reports success.)	`game.useMirror();`
	How many mirrors I'm carrying.	`game.getNumMirrorsHeld()`
	Seconds left on my active mirror (0 = none).	`game.getMirrorTimeRemaining()`
	Fills a 3-slot array with that item's location.	`game.getItemLoc(arr, 4);`
	0 = score, 1 = energy, 2 = mirror.	`game.getItemType(4)`

AsteroidBee MS- Light/Dark

Block	What it does	Generated C
	Energy level (0–5). The dropdown works for the opponent too.	`game.getEnergy()` / `game.getOtherEnergy()`
	Is that satellite in the Light Zone right now?	`game.checkInLight()` / `…Other()`
	Is that satellite in the Dark Zone right now?	`game.checkInDark()` / `…Other()`
	Seconds until the Light/Dark halves swap.	`game.getLightSwitchTime()`

AsteroidBee MS- Other

Block	What it does	Generated C
	Live score for either player (dropdown: `my` / `other`).	`game.getScore()` / `game.getOtherScore()`
	Thruster-seconds left (starts at 60).	`game.getFuelRemaining()`

How a whole program reads in C

The "C Code" toggle shows the generated file. A small program looks like this:

```
float myState[12];
int step;

void setPos(float x, float y, float z) {   // auto-generated helper
    float pos[3];
    pos[0] = x; pos[1] = y; pos[2] = z;
    api.setPositionTarget(pos);
}

//Begin page init
void init() {
    step = 0;
}
//End page init

//Begin page main
void loop() {
    api.getMyZRState(myState);
    if (step == 0) {
        setPos(0, -0.4, 0);
    }
}
//End page main
```

Global declarations come from the init page's "global variables" slot; everything in the main page's "loop" slot lands inside `void loop()`, which the satellite runs **once per second** for all 180 seconds.