Assignment 1: Hello, Satellite! — Teacher Guide

At a glance

Learning objective	Students create their first graphical project, learn the IDE layout (init page, main page, palette, workspace, C Code toggle, log), and discover for themselves that the main page runs once per second, by printing a message and counting the lines.
Time estimate	45–60 minutes
Project setup	Game: **Free Mode** · Editor: **Graphical Editor**
Prerequisite assignment	None
Blocks introduced	`DEBUG` (Debug category), the text block `""`

The one idea this assignment teaches

A graphical project has two pages of blocks, and they run on completely different schedules. The **init** page runs **once**, at the moment the match starts. The **main** page (its root block is labeled "loop") is re-read by the satellite **every single second** of the match. That is the whole lesson today. Instead of telling students this fact, we let them discover it. They will place one block that prints one message, press Simulate, and find dozens of copies of their message in the log. Their first reaction is usually "it's broken." It is not broken. It is the most important thing they will learn all season, demonstrating itself.

> ⏱ **Every second:** the satellite reads the main page from top to bottom and runs every block on it. Then, one second later, it reads the whole page again from the top. A single `DEBUG` block on the main page therefore prints once per second, every second, for the whole simulation. The blocks are not a story that plays once; they are a checklist the satellite re-reads every second.

Why does this matter so much? Almost every confusion students hit later in the season ("my satellite skipped my blocks," "it only did the last thing," "it took the picture before it aimed") comes from imagining blocks as a recipe that plays out over time. Today's repeating log message is that misconception in its simplest, safest form. Treat the surprise as the win: when a student can say "the main page ran my block 60 times because the page re-runs every second," they have the mental model the next six assignments are built on.

Before class

- [ ] Read how-blocks-run.md: five minutes, and it covers today's idea exactly.
- [ ] Make your own account on the Zero Robotics website and confirm you can log in.
- [ ] Do this assignment yourself once, start to finish (about 10 minutes). Create the project, place the `DEBUG` block, simulate, and look at the log so you know what students will see.
- [ ] Confirm student accounts exist, or budget 10 extra minutes at the start for account creation.
- [ ] Have these three screenshots ready to project: ../images/ide-new-project-dialog.png, ../images/ide-graphical-editor-layout.png, ../images/ide-log-output.png.
- [ ] Check that the lab machines can reach the Zero Robotics website in a browser (no software install is needed; everything runs in the browser).
- [ ] Print one copy of `printable.md` per student.

Walkthrough: what to say and do

1. **Log in.** Have everyone create an account (or log in) on the Zero Robotics website. Frame the story lightly: "You've just been hired as mission control for a spy satellite. Today's job: open a radio channel and prove the satellite is listening."

2. **Create the project.** Click **New Project**. Show ../images/ide-new-project-dialog.png on the projector. Three choices matter: give it a name (anything works, even "Hello Satellite"), choose the game **Free Mode**, and choose **Graphical Editor**. Say out loud: "Graphical, not Text. Text is typed code, and we're using blocks." Have everyone confirm both choices before clicking Create.

3. **Tour the editor.** Project ../images/ide-graphical-editor-layout.png and point at each region: the **two page tabs** (init and main), the **palette** of block categories on the left, the big empty **workspace** where blocks go, the **trash** (drag a block there to delete it), and the **"C Code" toggle** (ignore it for now; "we'll peek behind the curtain at the end"). Then show the two blocks that are already there and can't be deleted: the init page's root block (it has two slots, "global variables" and "init") and the main page's root block, labeled **"loop"**. Say it plainly: "The init page runs ONCE, the moment the match starts. The main page re-runs every second, all match long. Remember I said that. We're about to prove it."

4. **Build the program.** On the **main** page: open the **Debug** category in the palette and drag the `DEBUG` block into the "loop" slot. Then drag the text block `""` into the `DEBUG` block's socket. Click between the quote marks and type a message (for example, calling mission control). Warn them once: "Your words go BETWEEN the two quote marks. Don't delete the quotes."

5. **Save and simulate.** Click Save, then Simulate, and accept the **default settings**. Let the simulation finish.

6. **Open the log, and ask rather than tell.** Show ../images/ide-log-output.png and have students open their own log. The message repeats: one line per simulated second. Before explaining anything, ask the class: **"You placed ONE block. Why are there SIXTY lines?"** Let them argue for a minute. Someone will get there: the page must have run more than once. Then confirm it: the satellite reads the main page again every second, so the one block ran about 60 times, once per second. Have everyone actually count or estimate the lines; saying "about 60, because 60 seconds" out loud is the point of the day.

7. **The experiment: move the block.** Now drag the very same `DEBUG` block (with its text still attached) off the main page and into the **init** page's **"init"** slot. Ask for predictions FIRST: "How many lines this time?" Then simulate. The message prints **exactly once**, at the start. One block, two pages, two completely different behaviors, because the pages run on different schedules.

8. **Peek behind the curtain.** Flip the **"C Code"** toggle. Show that the blocks have become typed code: with the block on the main page, the message sits inside `void loop()`; with the block on the init page, it sits inside `void init()`. You don't need to explain C. Just say: "The blocks and this code are the same program in two costumes. The simulator calls `init()` once, then calls `loop()` once per second. That's why your message repeated." Toggle back to blocks.

The mistakes you will see

1. "My program is broken, it spammed the log!"

- **What it looks like:** A student placed one `DEBUG` block, simulated, and the log is full of dozens of copies of their message. They raise a hand and report a bug.
- **Why it happens:** They think of blocks as a story that runs once, start to finish. They expected their one block to print one line. This is the "blocks run in sequence over time" misconception in its simplest possible form, and surfacing it now, where nothing can go wrong, is the entire purpose of this assignment.

- **The question to ask:** "How many times did the satellite read your main page during a 60-second simulation?" Do not rush to answer it for them. When they say "sixty times?", the lesson has landed. **Treat the surprise as the win.** This student just learned the most important fact of the season.

2. The message is outside the quotes (or the quotes got deleted)

- **What it looks like:** The program fails to build, or the printed message is empty, even though the student "definitely typed it."
- **Why it happens:** The text block prints as `""`, and students type next to the quote marks instead of between them, or select-all and delete the quote marks along with the old text.
- **The question to ask:** "Is your message between the two quote marks?" If the quotes are gone, the quickest fix is to drag the text block to the trash and pull a fresh `""` from the Debug category.

3. Wrong editor or wrong game at project creation

- **What it looks like:** The palette looks wrong: different categories than the projected screenshot, or no block palette at all (a Text Editor project shows typed code instead of blocks).
- **Why it happens:** The New Project dialog has several choices, and a student picked Text Editor instead of Graphical Editor, or a game other than Free Mode.
- **The question to ask:** "When you created the project, which game and which editor did you pick?" The fix is painless: **just create a new project** with the right choices. Nothing is lost, and it takes under a minute.

Reference solution at a glance

The finished program is one block plus its text. The **init** page is untouched: both of its slots ("global variables" and "init") are empty. The **main** page has the `DEBUG` block sitting in the root "loop" slot, with the text block `""` plugged into its socket and the message calling mission control typed between the quotes. The step-7 variant is the same `DEBUG` block moved into the init page's "init" slot, leaving the main page empty. See reference-solution.md for the exact layout and the C code it generates.

Wrap-up questions

1. You placed one `DEBUG` block, but the log showed about 60 lines. So what is a block on the main page: a step in a story, or something else? (Looking for: a checklist item the satellite re-reads every second.)
2. If you wanted a message to appear exactly once at the start of the match, which page does the block go on, and why?
3. Predict before trying it: one `DEBUG` block saying "starting up" on the init page, and a different `DEBUG` block saying "still here" on the main page. What will the log look like? (One "starting up" line at the top, then "still here" once per second.)