

Virtual Rodent Behaviour Environment

User Manual

Running experiments on the RODENT website and in Google Colab

Python SDK version 0.4.2

October 11, 2026

Contents

Before you start	2
1 Running experiments on the website	4
1.1 Sign in	4
1.2 Protect your account with two-factor sign-in (optional)	4
1.3 The Projects page	5
1.4 Create a project	5
1.5 Add an experiment from a template	6
1.6 Your team and roles	8
1.7 The arena editor: draw your own arena	9
1.8 The workspace	12
1.9 Setup: conditions	12
1.10 Setup: arena	14
1.11 Setup: stimuli	16
1.12 Setup: templates and environment	16
1.13 Run	18
1.14 Review: results and replays	19
1.15 Use an experiment in Python	21
1.16 If something goes wrong on the website	22
2 Running experiments in Google Colab	23
2.1 Open the notebook and save your own copy	23
2.2 Store your RODENT login safely (one time only)	24
2.3 Run the notebook	24
2.4 Steps 1 and 2: install and sign in	26
2.5 Step 3: choose your experiment	26
2.6 Step 4: look at and change the setup	27
2.7 Step 5: run	27
2.8 Step 6: results	29
2.9 Steps 7 and 8: save to the website and download the data	30
2.10 Doing more	30
2.11 Coming back later	31
2.12 If something goes wrong in Colab	31
3 Training your own rat model	32
3.1 Steps 1 to 3: set up, sign in and choose the experiment	32

3.2	Step 4: explore the experiment	33
3.3	Step 5: define the task	33
3.4	Step 6: the model	34
3.5	Step 7: train	34
3.6	Step 8: results	35
3.7	Step 9: save to your project and find it on the website	37
3.8	Step 10: use the model in another experiment	38
3.9	Good to know	39

Before you start

RODENT simulates a rat moving through a behavioural test arena, such as a maze or an open field. You set up the arena and the conditions (doors, lights, sounds, odours and treatment), run the simulation, and study where the rat went and for how long.

You can use RODENT in two ways, and they share the same projects:

- **On the RODENT website** (Section 1): you watch the rat move in 3D, build and edit arenas visually, manage your team, and replay saved runs.
- **In Google Colab** (Section 2): you run experiments from a prepared notebook by pressing **Run**. There are no 3D graphics, runs are fast, and the results arrive as tables, charts and files you can analyse.

In Colab you can also **train your own model** to control the rat, and share it with your project (Section 3).

Anything you create or save in one place appears in the other.

What you need

- A computer with an internet connection and a modern web browser (Chrome, Edge, Firefox or Safari).
- A **RODENT account**: an email address and password for the RODENT website (Section 1.1).
- For Colab only: a **Google account** (for example a Gmail address).

You do not need to install anything on your computer, and you do not need to know how to program.

How to read the screenshots

Most screenshots have red boxes with numbers such as **1** and **2**. The text next to each screenshot explains each number in order.

Words used in this manual

Project	A study. It contains experiments and has a team of people.
Experiment	One arena setup inside a project: the layout, doors, stimuli, treatment and seed.
Arena	The space the rat moves in, divided into named <i>regions</i> such as “Central Arena” or “Dark Shelter”.
Stimulus	A light, sound or odour source that can be on or off.
Seed	A number that fixes the random choices in a run. The same experiment with the same seed always produces exactly the same run.
Step	One tick of the simulation clock. By default one step is 0.1 seconds, so 5000 steps is 500 seconds (about 8 minutes) of simulated time.
Run	One simulation of an experiment, from start to finish.
Replay	A saved run that can be played back on the website.
Draft, published	A <i>draft</i> experiment can still be changed. A <i>published</i> version is frozen, so that results can always be traced to an exact setup.
Role	What you may do in a project (see Section 1.6).

Important

The built-in rat is moved by a simple statistical model, which is a placeholder. You can now replace it with a model you train yourself (Section 3), but neither is a validated biological result. Use RODENT results to explore and compare setups, not as evidence about real animals.

1 Running experiments on the website

1.1 Sign in

Open `https://rodent-tau.vercel.app/` in your browser.

- 1 **Email:** the email address of your RODENT account.
- 2 **Password.**
- 3 **Sign in.**
- 4 **Create an account:** if you do not have an account, enter your email and a password twice and click **Create researcher account**. If you receive a confirmation email, open its link first, then sign in.
- 5 **Request password reset:** if you forgot your password, type your email, click this, and follow the link in the email (check your spam folder too).

Important

Create only one account per person. If signing in fails, reset your password rather than creating a second account.

1.2 Protect your account with two-factor sign-in (optional)

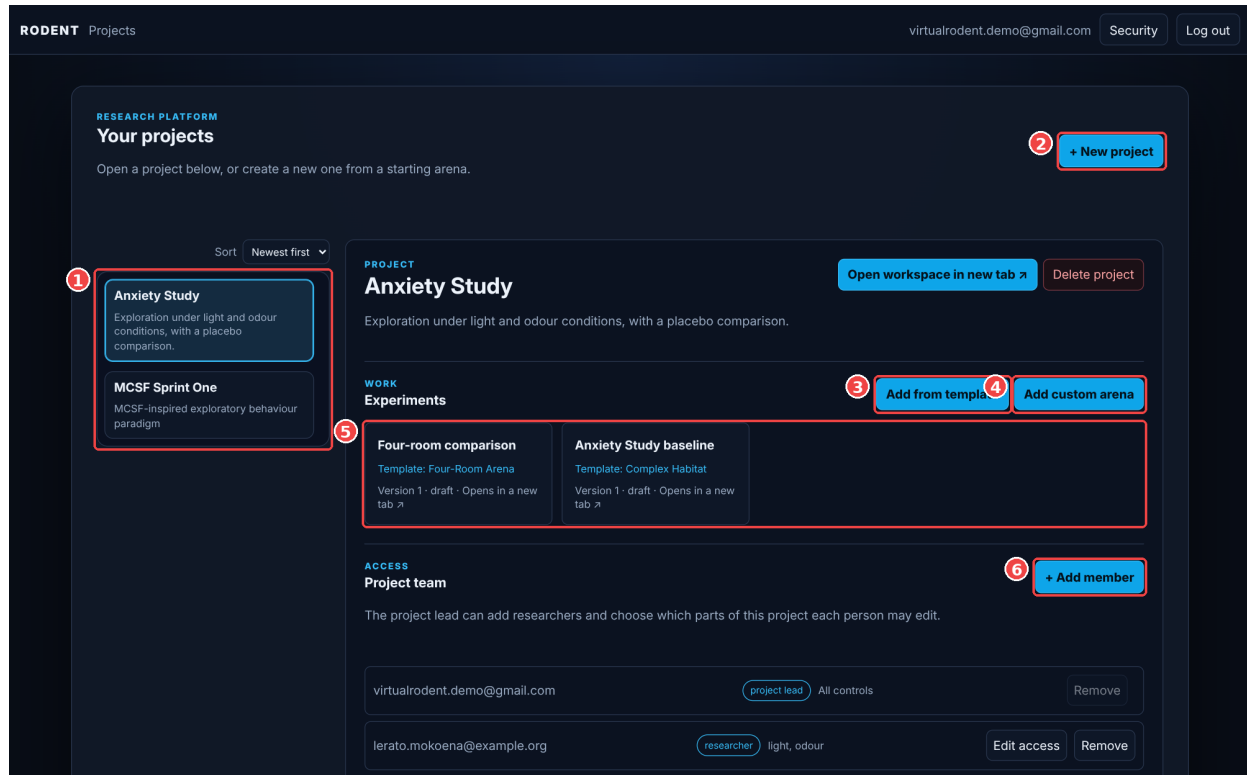
Click **Security** at the top right, then **Set up two-factor authentication**. Scan the QR code with an authenticator app (Google Authenticator, Microsoft Authenticator, Authy or similar), type the six-digit code it shows, and click **Verify and enable**. From then on the website asks for a code from the app each time you sign in.

Tip

Two-factor sign-in applies to the website. The Python notebook (Section 2) still signs in with your email and password only, so keep your password private.

1.3 The Projects page

After signing in you see your projects.



- 1 **Your projects:** click a project to open it on the right. **Sort** changes the order.
- 2 **+ New project:** start a new study.
- 3 **Add from template:** add an experiment from a ready-made arena (project leads only).
- 4 **Add custom arena:** draw your own arena in the arena editor (project leads only, Section 1.7).
- 5 The project's experiments. Click one to open it. It opens in a **new browser tab**, called the *workspace*, and the Projects page stays open in the original tab.
- 6 **+ Add member:** add people to the project team (Section 1.6).

1.4 Create a project

Click **+ New project**. In the window that opens:

NEW PROJECT
Create a new project

A project starts with one baseline experiment. Add more from a template, or draw your own in the arena editor.

Project name:

Baseline arena:

Description:

Project modules:

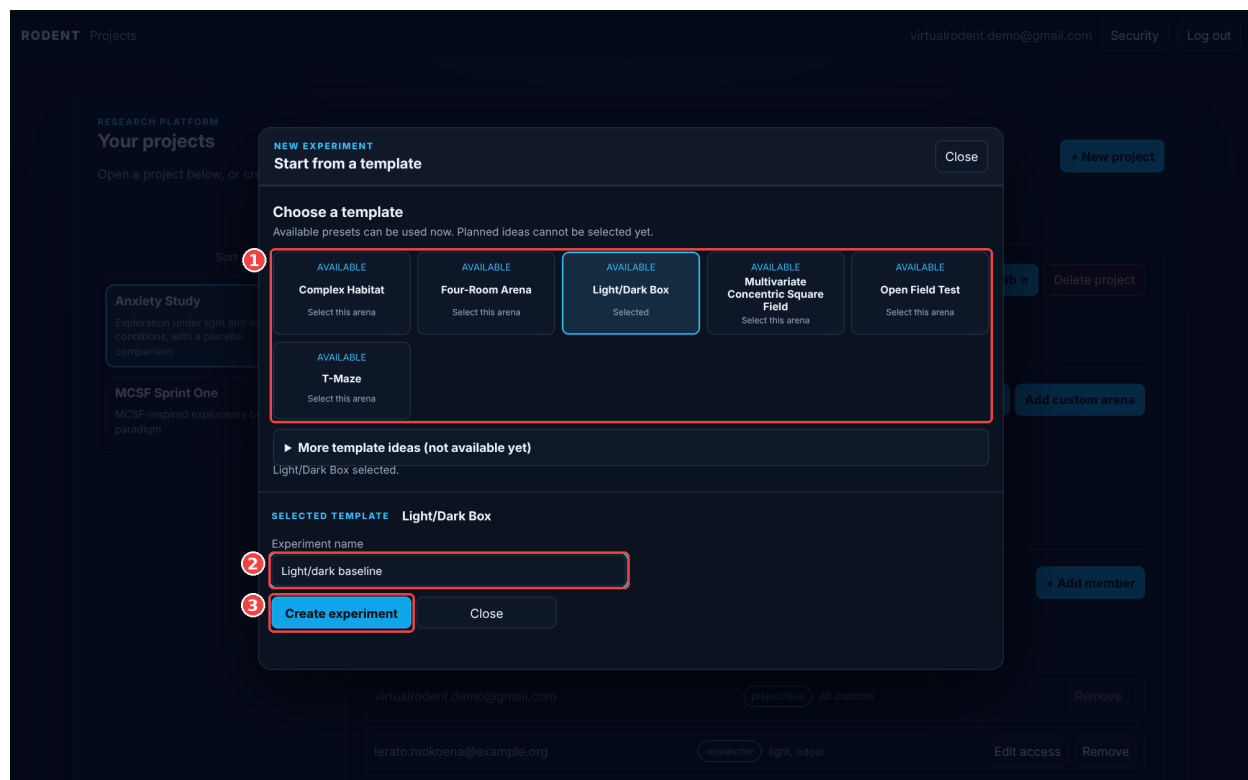
- 1 **Project name:** a name you will recognise. Use a name that no other project of yours has.
- 2 **Baseline arena:** the arena the project starts with.
- 3 **Description:** optional notes about the study.
- 4 **Project modules:** what may be customised in this project: **All modules**, only light, sound or odour, or **Custom modules** to choose.
- 5 **Create project.** You become the project's **lead**, and a baseline experiment is created.

Tip

A new project's baseline starts without lights, sounds or odours, so that you add exactly the stimuli your study needs (Section 1.11).

1.5 Add an experiment from a template

Click **Add from template**.

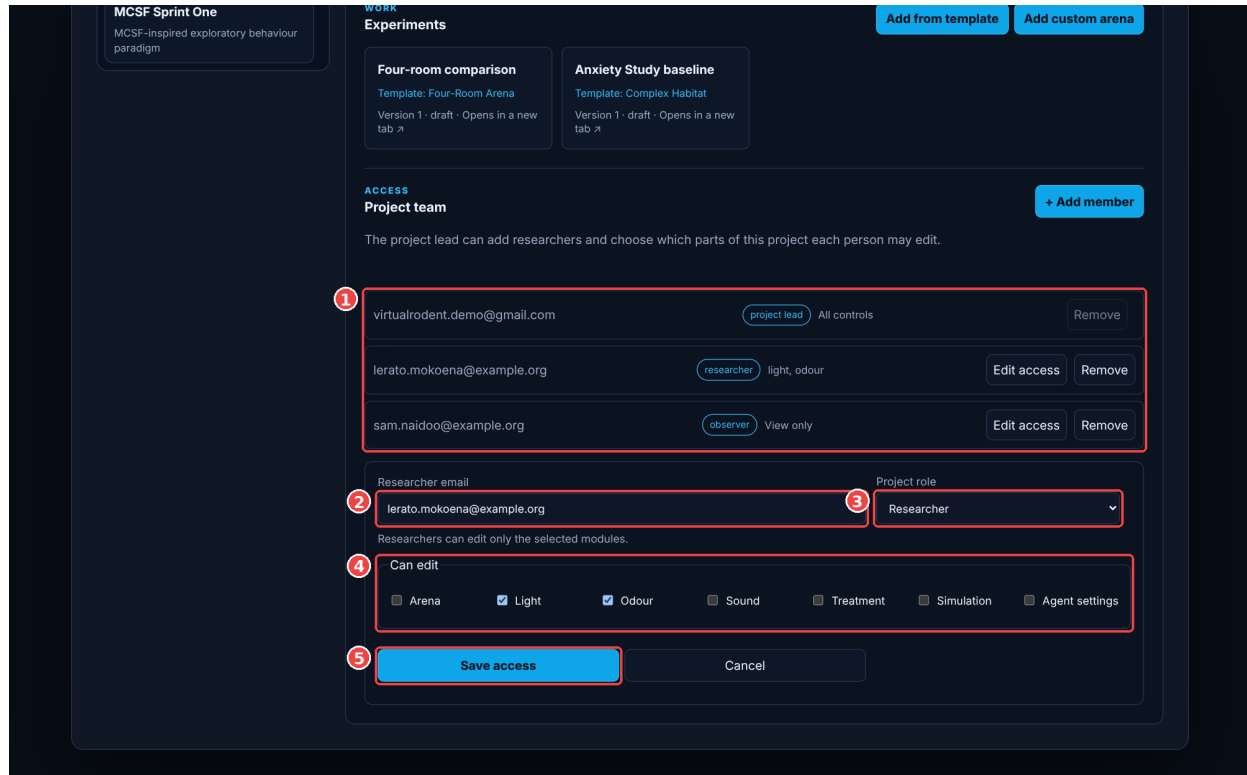


- 1 Click the arena you want. The available arenas are listed in the table below.
- 2 **Experiment name:** a name for the new experiment.
- 3 **Create experiment.**

Complex Habitat	A large arena of rooms and corridors, with a dark shelter, a foraging zone and a bridge route.
Four-Room Arena	Four rooms joined by doorways.
Light/Dark Box	A lit room and a dark room, joined by an opening.
Multivariate Concentric Square Field	The MCSF, with a raised bridge reached by ramps.
Open Field Test	One open square.
T-Maze	A T-shaped maze with a choice between two arms.

1.6 Your team and roles

The project lead manages the team under **Project team**.

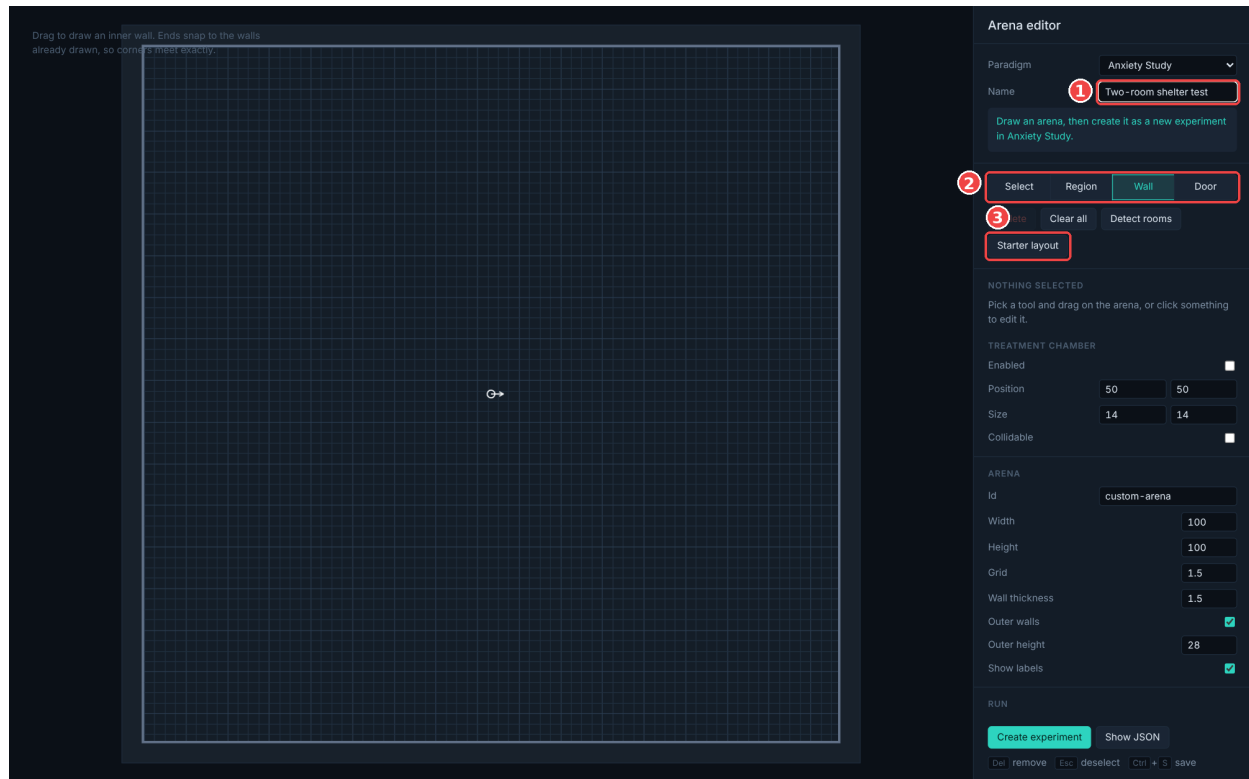


- 1 The team. Each person shows their role and what they may edit. Use **Edit access** or **Remove** to change someone.
- 2 **Researcher email**: the person must already have a RODENT account.
- 3 **Project role**: see the table below.
- 4 **Can edit**: for researchers, tick the modules they may change. Here the researcher may change only light and odour.
- 5 **Add to project** (or **Save access** when editing someone).

Project lead	Everything: edit all enabled modules, add experiments from templates or the arena editor, manage the team, publish versions.
Researcher	Edits only the modules ticked under Can edit . Can run experiments, save runs, and copy experiments into new drafts.
Observer	Can look at the project and its results, but cannot change or run anything.

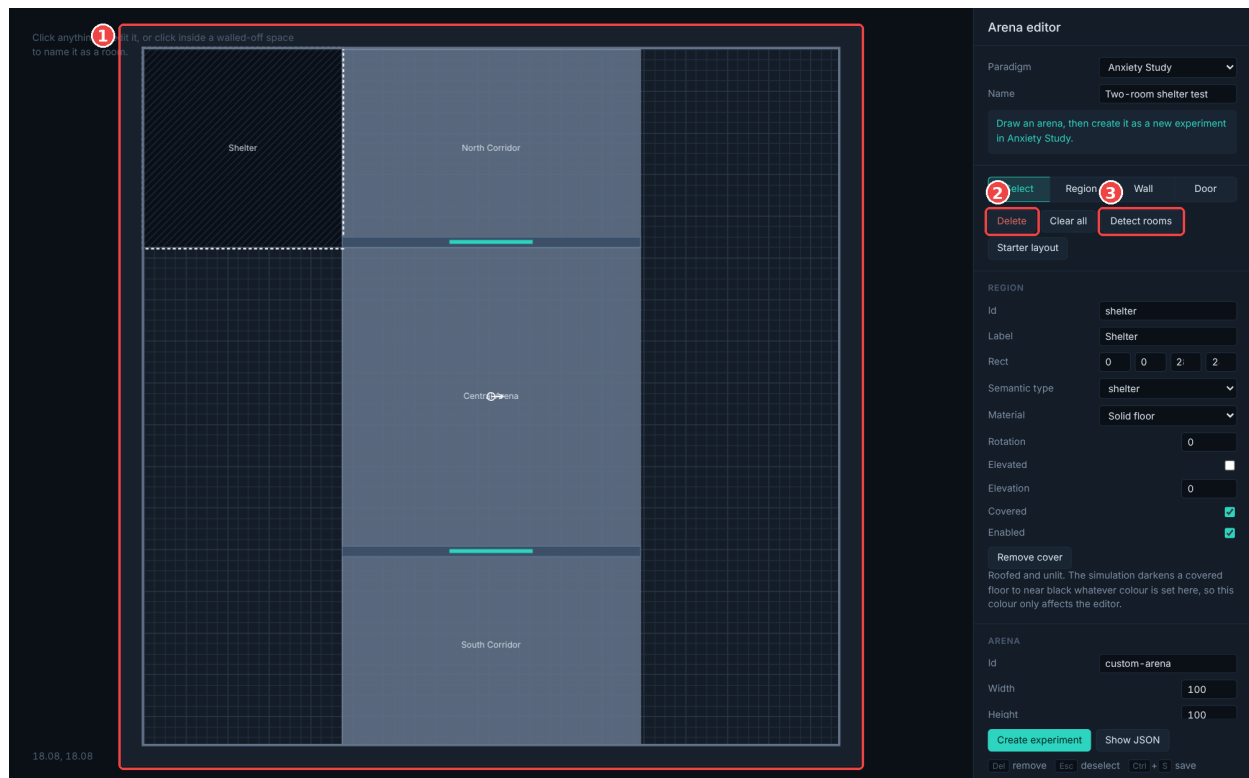
1.7 The arena editor: draw your own arena

Add custom arena opens the arena editor in a new tab. You draw the layout from above, then create it as a new experiment in the project.



- 1 **Name:** the name of the new experiment. **Paradigm** above it is the project it will be added to.
- 2 The drawing tools:
 - **Select:** click anything to edit it, or click inside a walled-off space to name it as a room.
 - **Region:** drag to draw a floor region (an area the results report time for).
 - **Wall:** drag to draw an inner wall. Its ends snap to walls already drawn, so corners meet exactly.
 - **Door:** click on a wall to cut a doorway through it.
- 3 **Starter layout:** fills the arena with a simple example (a central arena, two corridors and a shelter) that you can change. **Clear all** empties the arena.

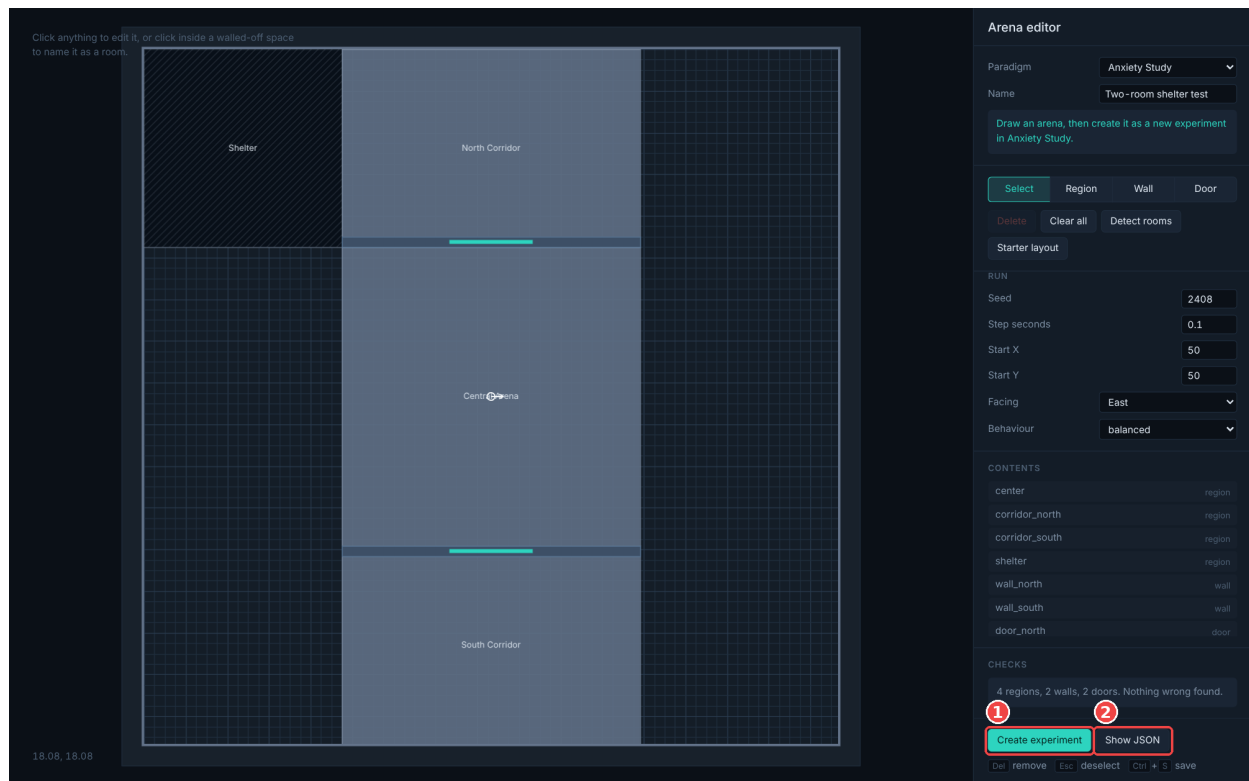
Editing what you drew. With **Select**, click a region, wall or door. Its settings appear on the right.



- 1 The selected object is outlined. For a region you can change its **Label** (the name in results), **Rect** (position and size), **Semantic type**, **Material**, **Rotation**, and whether it is **Elevated**, **Covered** or **Enabled**. **Make dark room** turns a region into a covered, dark shelter.
- 2 **Delete**: removes the selected object (or press `Del`).
- 3 **Detect rooms**: finds every space enclosed by walls that does not have a region yet, and creates one for it.

Further down the panel:

- **Treatment chamber**: switch it on and set its position and size.
- **Arena**: the arena's **Width** and **Height** in centimetres, the **Grid** that drawing snaps to, **Wall thickness**, and the **Outer walls** and their height.
- **Run**: the **Seed**, **Step seconds**, where the rat starts (**Start X**, **Start Y**), which way it faces (**Facing**) and its **Behaviour** (balanced, explorer or shelter seeker).



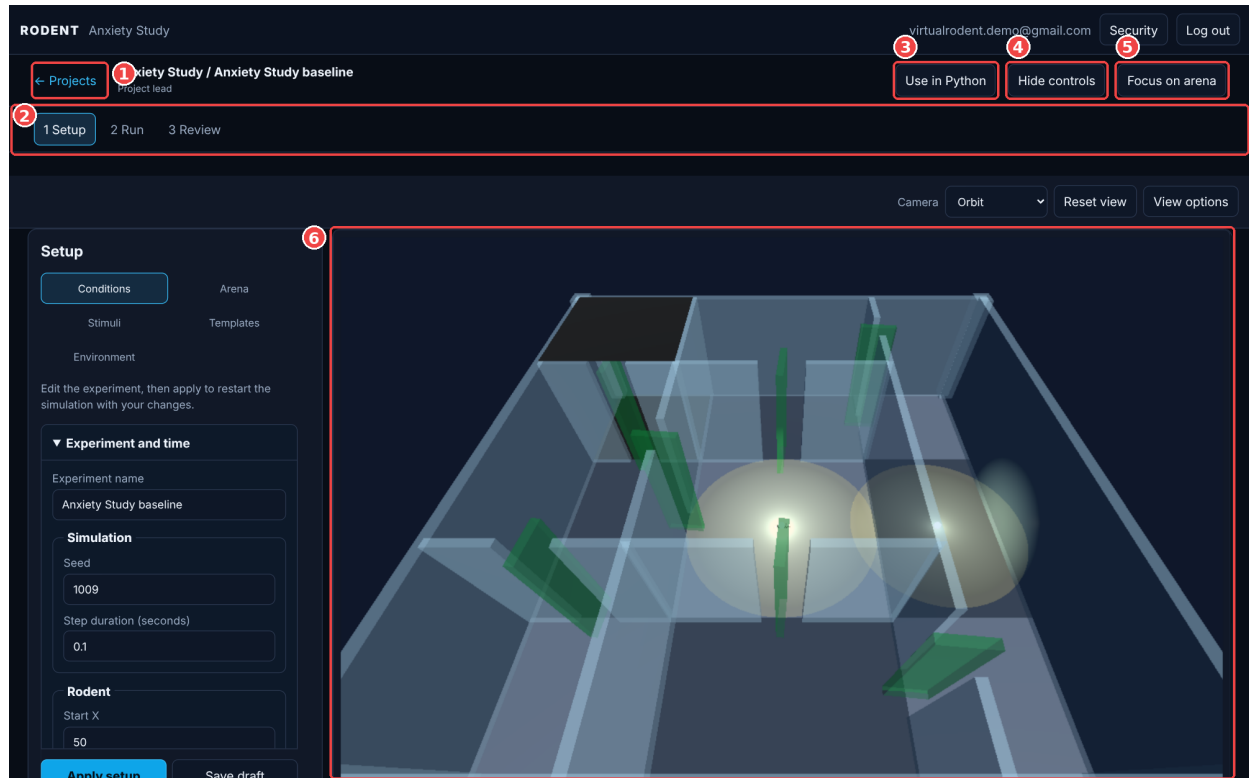
At the bottom, **Contents** lists everything in the arena and **Checks** reports problems, for example a room without a region, or a region the rat cannot reach. When the checks say *Nothing wrong found*:

- 1 **Create experiment:** adds the arena to the project as a new draft experiment (or press `Ctrl+S`).
- 2 **Show JSON:** shows the arena as a file, for advanced users.

Press `Esc` to deselect. The new experiment appears on the Projects page; open it to add stimuli and conditions.

1.8 The workspace

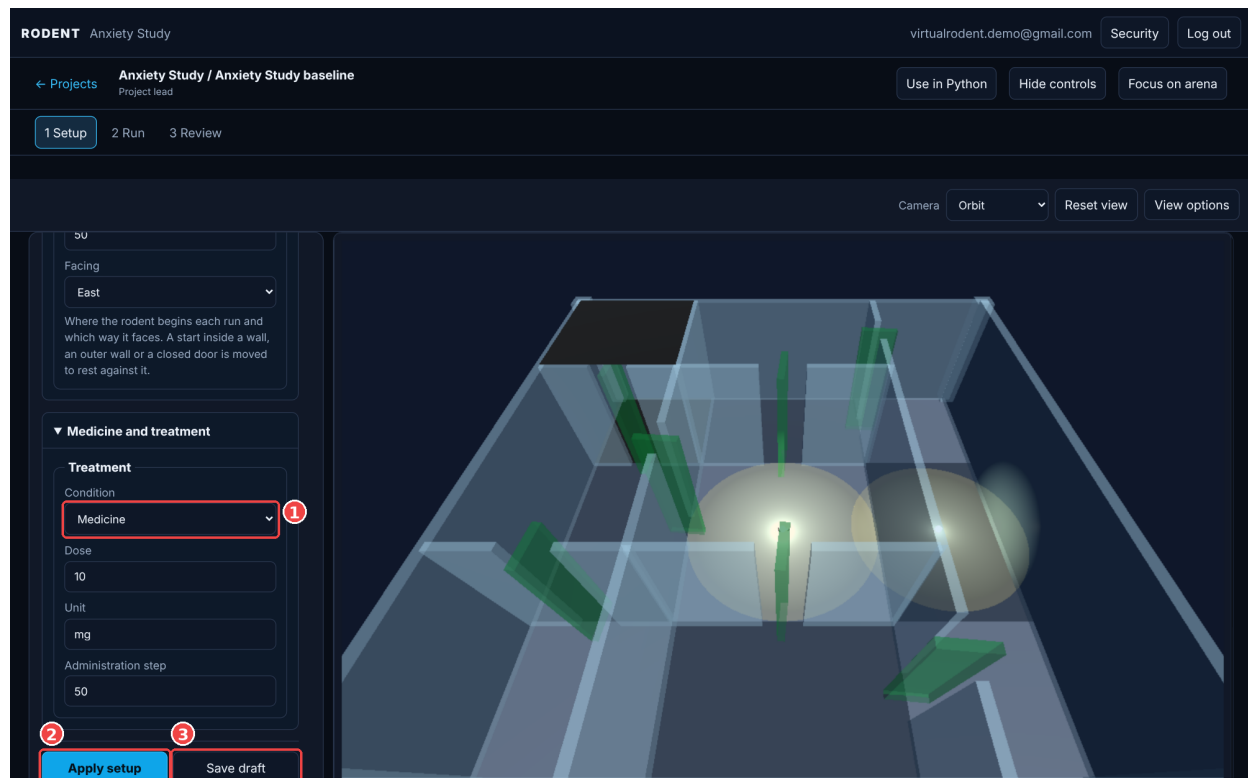
Opening an experiment shows its workspace: the 3D arena and three stages.



- 1 ← **Projects**: back to the project list.
- 2 The three stages: **1 Setup** (change the experiment), **2 Run** (run it and watch the rat), **3 Review** (results, downloads and replays).
- 3 **Use in Python**: run this experiment from Google Colab (Section 1.15).
- 4 **Hide controls**: makes the arena larger.
- 5 **Focus on arena**: shows only the arena; press `Esc` or **Exit focus view** to return.
- 6 The 3D arena. Drag with the left mouse button to rotate, drag with the right mouse button to move sideways, and use the mouse wheel to zoom.

1.9 Setup: conditions

Under **1 Setup**, the **Conditions** tab holds the experiment's name, **Seed** and **Step duration**, where the rat starts and which way it faces, and the treatment.



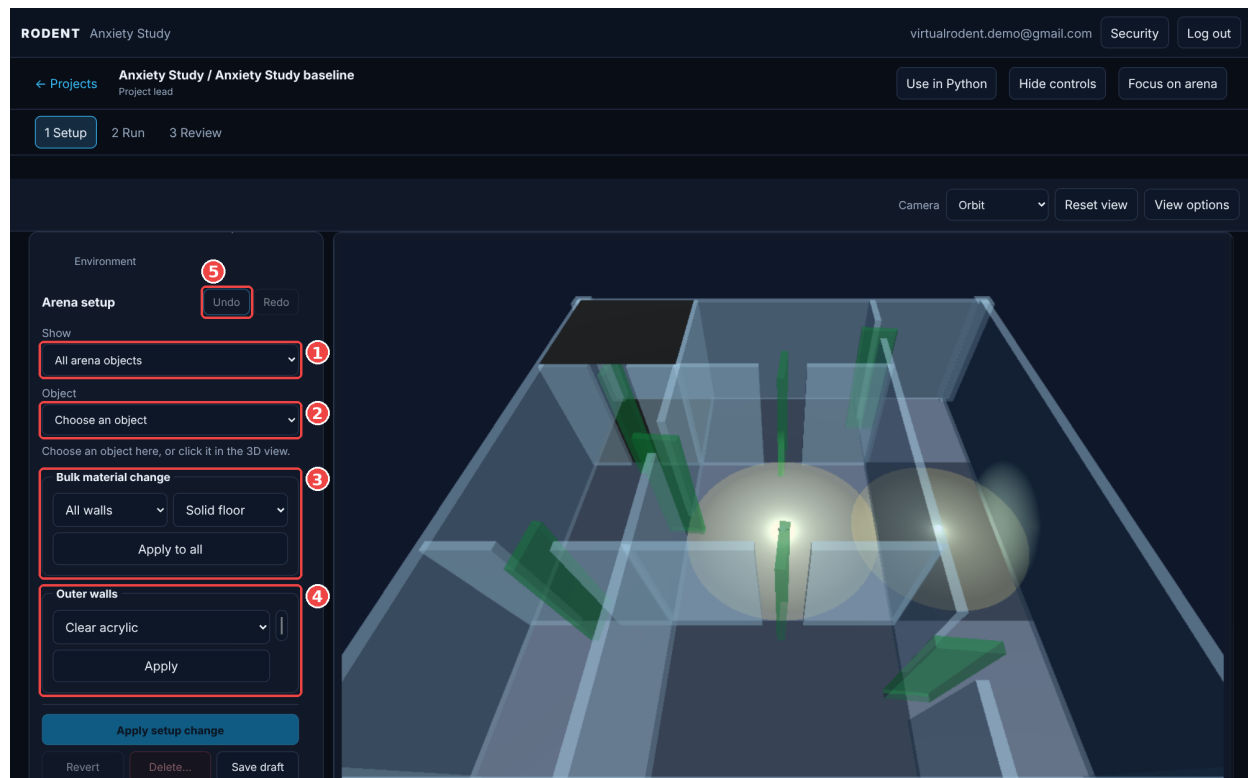
- 1 **Medicine and treatment: Condition** (No treatment, Placebo or Medicine), **Dose**, **Unit** and **Administration step** (the step at which it is given).
- 2 **Apply setup**: shows your changes in the 3D arena and restarts the simulation from the seed. Nothing is stored yet.
- 3 **Save draft**: stores the changes, so everyone in the project (and the Colab notebook) sees them.

Important

Changes you only *apply* are lost when you close the tab. Click **Save draft** to keep them.

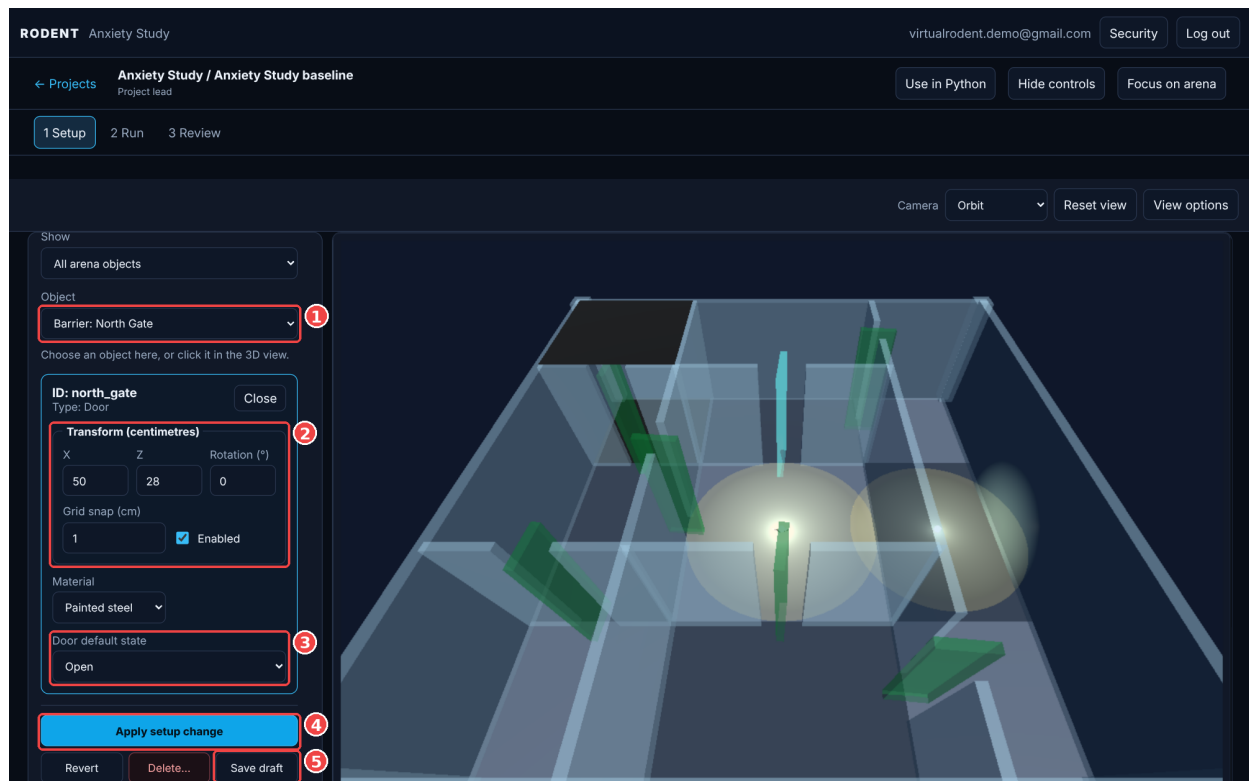
1.10 Setup: arena

The **Arena** tab edits the layout: floors, walls, doors and the treatment chamber.



- 1 **Show:** list only walls, doors, floors and so on.
- 2 **Object:** choose an object, or **click it in the 3D view**.
- 3 **Bulk material change:** give all walls, all floor regions or all doors the same material, then click **Apply to all**.
- 4 **Outer walls:** change the material and colour of the arena's surrounding walls.
- 5 **Undo** and **Redo:** step back and forward through your arena changes.

When an object is chosen, its settings open:



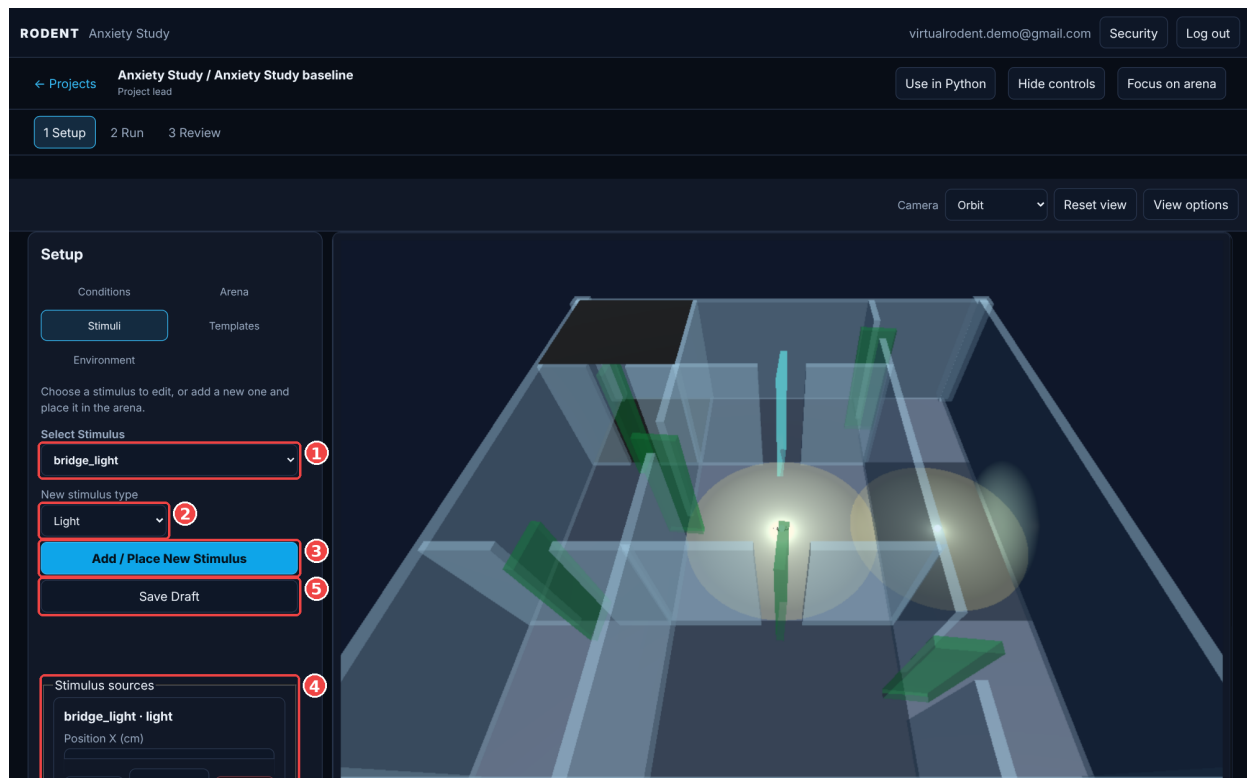
- 1 The chosen object (here the North Gate door). It is highlighted in the 3D view.
- 2 **Transform (centimetres):** position (**X**, **Z**) and **Rotation (°)**. **Grid snap (cm)** rounds positions to neat values. Untick **Enabled** to leave the object out of the run without deleting it.
- 3 Settings for this kind of object, here the **Door default state** (Open, Closed or Locked). **Material** and colour are above it.
- 4 **Apply setup change:** the 3D arena updates straight away. **Revert** undoes changes you have not applied, and **Delete...** removes the object (**Undo** can restore it).
- 5 **Save draft:** stores the arena.

Tip

If a change is refused, a message in red explains why, for example a wall placed outside the arena. Nothing is changed in that case.

1.11 Setup: stimuli

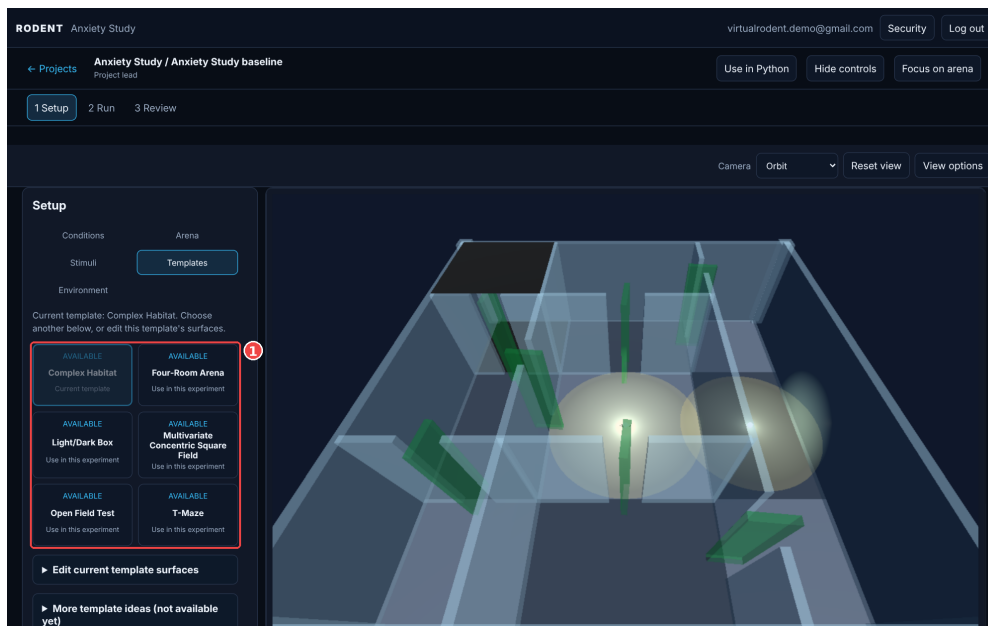
The **Stimuli** tab adds and edits lights, odours and sounds.



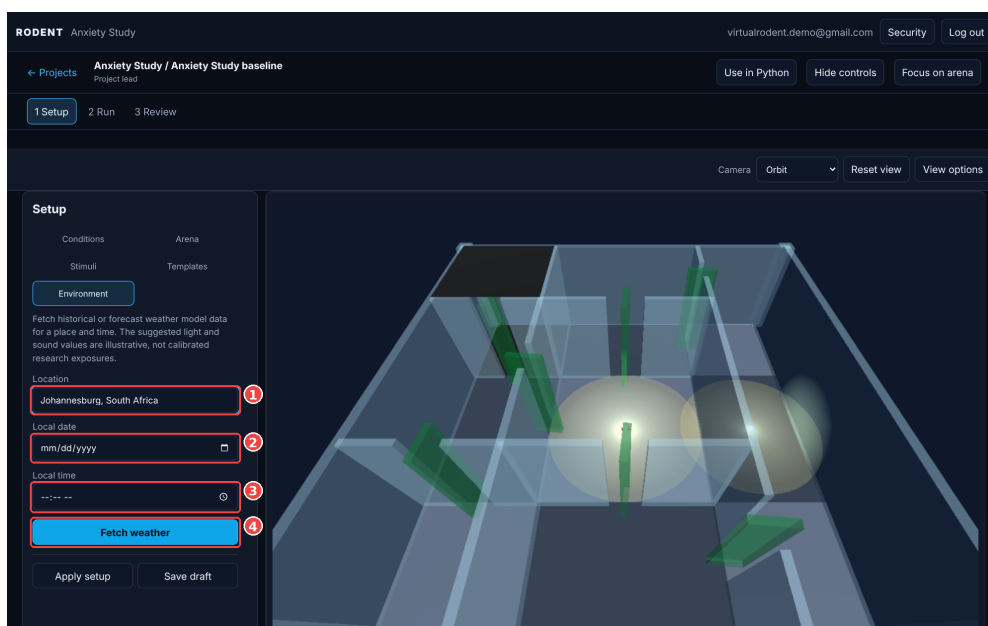
- 1 **Select Stimulus:** choose an existing stimulus to edit.
- 2 **New stimulus type:** Light, Odour or Sound.
- 3 **Add / Place New Stimulus:** adds one, then click in the 3D arena where it should go.
- 4 **Stimulus sources:** the chosen stimulus's settings: position (**Position X**, **Position Z**), **Active at start**, **Radius (cm)**, its strength (brightness in lux for light, volume for sound, intensity for odour), and **Activate** at the start or at a chosen step. Click **Apply** to update the arena, **Reposition in Arena** to move it by clicking, or **Delete** to remove it (this also removes protocol events that use it).
- 5 **Save Draft:** stores the stimuli.

1.12 Setup: templates and environment

Templates replaces the whole experiment with a different ready-made arena 1. This needs full project access; afterwards, click **Apply setup** and **Save draft**. **Edit current template surfaces** changes the floors of the current arena.



Environment suggests an ambient light and sound level from real weather.

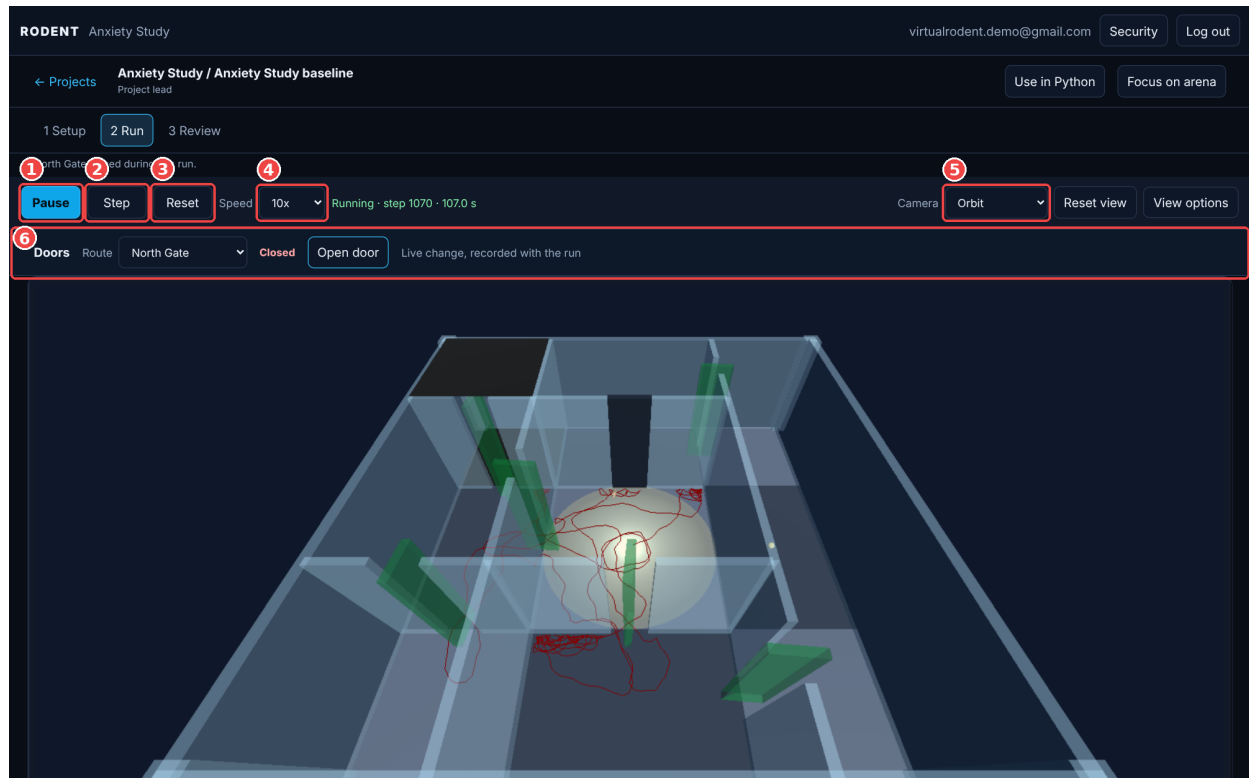


- 1 **Location:** a place, for example *Johannesburg, South Africa*.
- 2 **Local date.**
- 3 **Local time.**
- 4 **Fetch weather.** Check the suggestions, then click **Use suggestions in draft**, **Apply setup** and **Save draft**.

The suggested values are illustrative, not calibrated research exposures.

1.13 Run

Click **2 Run**.



- 1 **Start / Pause:** start the simulation, or pause it. The status next to the buttons shows the step number and the simulated time.
- 2 **Step:** advance a single step while paused.
- 3 **Reset:** restart the run from the beginning.
- 4 **Speed:** how fast the run is *shown*: **1x**, **10x**, **100x** or **Max**. It does not change the result.
- 5 **Camera:** **Orbit** (rotate around the arena), **Free fly**, or **Follow agent** (the camera follows the rat). **Reset view** returns to the start view, and **View options** shows or hides labels and adjusts the mouse.
- 6 **Doors:** open or close a door *during* the run. Choose it under **Route** and click **Close door** or **Open door**. The run continues, and the change is recorded in the data at that moment. Here the North Gate was closed during the run. This needs permission to edit the arena.

The red line in the arena is the path the rat has taken so far.

1.14 Review: results and replays

Click **3 Review** after a run.

RODENT Anxiety Study virtualrodent.demo@gmail.com Security Log out

[← Projects](#) **Anxiety Study / Anxiety Study baseline** Project lead Use in Python Hide controls Focus on arena

1 Setup 2 Run **3 Review**

Replay saved to this experiment.

Review
Inspect the run, download results or play a saved replay.

Recorded steps	1210
Duration	121.0 s
Seed	1009
Treatment	medicine
Regions: center 654 · corridor_s 314 · corridor_w 216 · foraging 26	
Distance travelled	1082.1 cm
Mean movement speed	8.94 cm/s
Steps affected by stimuli	534
Stimulus exposure: bridge_light 0.000 mean, 0 steps · center_light 0.048 mean, 534 steps · dark_odor 0.000 mean, 0 steps · foraging_tone 0 mean, 0 steps	

Download JSON Download CSV Read JSON in browser

Saved runs

Save this run

10/9/2026, 1:22:34 AM · seed 1009 · Play →

10/9/2026, 1:13:24 AM · seed 1009 · Play →

- 1 The **summary**: recorded steps, duration, seed, treatment, time in each region, distance travelled, speed, and how much the stimuli affected the rat.
- 2 **Download CSV**: the step-by-step table, for Excel, Google Sheets, R, SPSS or Python.
- 3 **Download JSON**: the complete run. **Read JSON in browser** shows it on screen.
- 4 **Save this run**: stores the run so it can be replayed later.
- 5 **Saved runs**: every saved run, with its date and seed. This includes runs saved from the Colab notebook.

Replaying a run. Click a saved run to load it.

The screenshot displays the RODENT Anxiety Study interface. At the top, the title "RODENT Anxiety Study" is visible, along with a user email "virtualrodent.demo@gmail.com" and buttons for "Security" and "Log out". Below the title, there's a navigation bar with "← Projects" and "Anxiety Study / Anxiety Study baseline". A "Project lead" label is also present. On the right, there are buttons for "Use in Python", "Hide controls", and "Focus on arena".

The main interface is divided into two main sections. On the left, there's a "Review" panel. It contains a "Replay loaded. Press Play to start." message. Below this, there's a "Review" section with the following data:

- Recorded steps: 1210
- Duration: 121.0 s
- Seed: 1009
- Treatment: medicine
- Regions: center 654 · corridor_s 314 · corridor_w 216 · foraging 26
- Distance travelled: 1082.1 cm
- Mean movement speed: 8.94 cm/s
- Steps affected by stimuli: 534

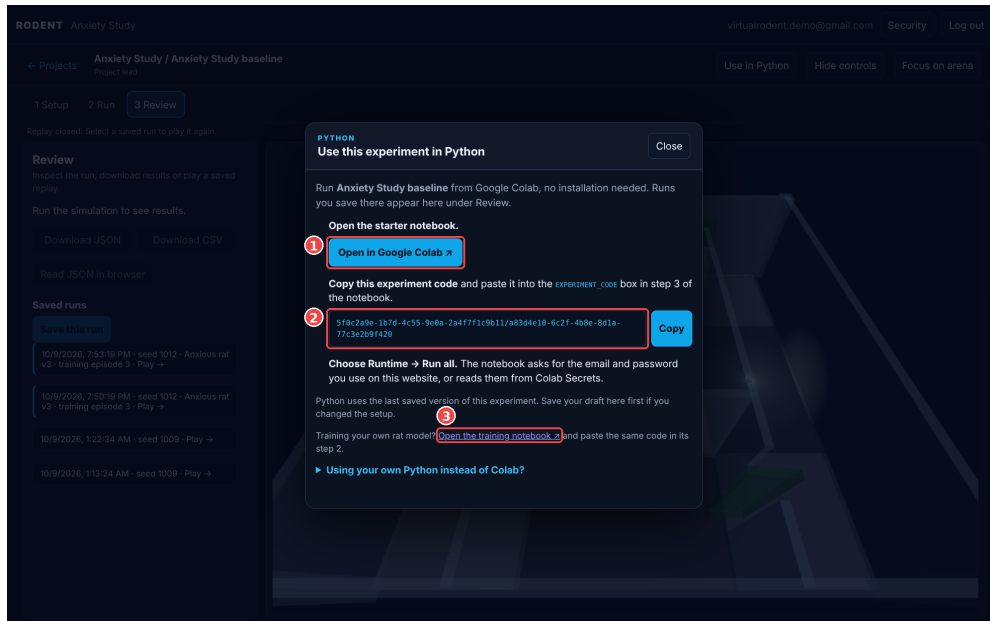
Below the data, there's a "Download JSON" button, a "Download CSV" button, and a "Read JSON in browser" button. At the bottom of the left panel, there's a "Saved runs" section with a "Save this run" button and two entries for runs saved on 10/9/2026.

The right section shows a 3D arena with a central light source and several green rectangular obstacles. A red line indicates the path of the rodent. At the bottom of the arena, there's a "Saved replay" control panel. It includes a "Frame 40 of 1210" indicator, a "Timeline" slider, and buttons for "Play", "Restart", and "Exit replay". The "Play" button is highlighted with a red circle and the number 1. The "Restart" button is highlighted with a red circle and the number 2. The "Timeline" slider is highlighted with a red circle and the number 3. The "Exit replay" button is highlighted with a red circle and the number 4.

- 1 **Play:** plays the run back in 3D.
- 2 **Restart:** back to the beginning.
- 3 **Timeline:** drag to jump to any step.
- 4 **Exit replay:** return to the live arena.

1.15 Use an experiment in Python

Use in Python, at the top of the workspace, connects an experiment to the Colab notebook.



- 1 **Open in Google Colab:** opens the starter notebook (Section 2).
- 2 The **experiment code**: it tells the notebook exactly which experiment to use. **Copy** copies it; paste it into the `EXPERIMENT_CODE` box in step 3 of the notebook.
- 3 **Open the training notebook:** for training your own rat model on this experiment (Section 3). Paste the same code into its step 3.

Python uses the last *saved* version of the experiment, so click **Save draft** first if you changed the setup. **Using your own Python instead of Colab?** shows ready-made code for people who use Python on their own computer.

1.16 If something goes wrong on the website

Problem	What to do
The arena stays on the loading screen	Wait up to a minute on the first visit while the simulator downloads, then reload the page (<code>Ctrl</code> + <code>F5</code> , or <code>Cmd</code> + <code>Shift</code> + <code>R</code> on a Mac).
A control is greyed out	Your role does not allow that change, or the run is still going: pause it first.
Changes disappeared after re-opening	They were applied but not saved. Remember Save draft .
The experiment will not change	It may be published (frozen). Ask the project lead, or work on a copy.
A new tab did not open	Your browser blocked it. Allow pop-ups for the website, or the experiment opens in the same tab instead.
The arena editor will not create the experiment	Give it a name, and fix what Checks reports.
A saved run from Colab is missing	Reload the Review stage. Check you opened the same project and experiment.

For anything else, contact your project lead.

2 Running experiments in Google Colab

Google Colab is a free Google service that runs prepared steps (called *cells*) in your web browser. The RODENT starter notebook is a page of such cells: you press **Run all**, and each cell does one part of the work.

2.1 Open the notebook and save your own copy

1. Open the notebook: either click **Use in Python** → **Open in Google Colab** in the website's workspace (Section 1.15), or open this link: https://colab.research.google.com/github/HuzaifaMajeed/rodent-sdk-releases/blob/main/notebooks/getting_started.ipynb
2. If Google asks you to sign in, sign in with your Google account.
3. In the menu at the top, click **File**, then **Save a copy in Drive**.
4. A new tab opens with *your* copy, named “Copy of getting_started”. **Use this copy from now on.** It is saved in your Google Drive, in a folder called **Colab Notebooks**, so you can find it again later.

S01 (your existing file)

Tip

Saving a copy matters: the shared notebook comes from the RODENT GitHub page and is read-only, so your settings would not be kept without your own copy.

2.2 Store your RODENT login safely (one time only)

The notebook signs in to RODENT for you. So that your password never appears on the page, you store it in Colab's *Secrets*, a private safe attached to your Google account.

1. In the narrow bar on the left edge of the notebook, click the **key icon (Secrets)**.
2. Click **Add new secret**. In **Name**, type exactly `RODENT_EMAIL`. In **Value**, type the email address you use for the RODENT website.
3. Click **Add new secret** again. Name: `RODENT_PASSWORD`. Value: your RODENT website password.
4. Switch on **Notebook access** for **both** secrets (the switch next to each one).

S02 (your existing file)

Important

The names must be typed exactly as shown, in capital letters with the underscore. Secrets belong to your Google account, so you only do this once. If you skip this step, the notebook asks you to type your email and password instead.

2.3 Run the notebook

1. First fill in step 3 of the notebook (Section 2.5).
2. In the menu at the top, click **Runtime**, then **Run all**.
3. If Colab shows “*Warning: This notebook was not authored by Google*”, click **Run anyway**. This appears for any notebook written by someone other than Google.
4. If Colab asks whether the notebook may use your secrets, click **Grant access**.

803 (your existing file)

The cells run one after another, from top to bottom. A spinning circle next to a cell means it is working; a green tick means it has finished. Results appear underneath each cell. The **first run takes about a minute**, because the RODENT simulator is downloaded into Colab. Later runs in the same session are much faster.

To run a single cell again, click the **play button** to its left, or click inside it and press `Shift+Enter`.

2.4 Steps 1 and 2: install and sign in

1. Install

```
%pip install --quiet rodent-sdk==0.4.2
import rodent_sdk as rodent
```

Step 1 installs RODENT (the `rodent-sdk` package) into Colab. Its version is fixed, so the same notebook always gives the same results.

2. Sign in

```
client = rodent.login()
print("Your projects:")
for project in client.projects():
    print(" - ", project.name)
```

```
Your projects:
- Anxiety Study
- Shared Study
```

Step 2 signs in and lists your projects.

2.5 Step 3: choose your experiment

3. Choose your experiment

On the website, open your experiment, click **Use in Python** and copy the code into `EXPERIMENT_CODE`.

Or leave it empty and type the project and experiment names instead.

```
EXPERIMENT_CODE = "5f0c2a9e-1b7d-4c55-9e0a-2a4f7f1c9b11/a83d4e10-6c2f-4b8e-8d1a-77c3e2b9f420" #@param {type:"string"}
PROJECT = "" #@param {type:"string"}
EXPERIMENT = "" #@param {type:"string"}

if EXPERIMENT_CODE:
    PROJECT, EXPERIMENT = EXPERIMENT_CODE.split("/")

exp = client.project(PROJECT).experiment(EXPERIMENT)
print("Using:", exp.name)
```

```
Using: Anxiety Study baseline
```

No project yet? (optional)

Type a name, pick an arena and run this cell. Then go back to step 3 and use the new project name. Its experiment is called *your project name* + `baseline`.

```
NEW_PROJECT = "" #@param {type:"string"}
ARENA = "complex_habitat" #@param ["complex_habitat", "mcsf_sprint_one", "open_field", "light_dark_box", "t_maze", "four_rc"]

if NEW_PROJECT:
    client.create_project(NEW_PROJECT, template=ARENA)
    print("Created:", NEW_PROJECT)
```

In Colab, the three settings below appear as boxes on the right of the cell:

EXPERIMENT_CODE The easiest way. Paste the code from **Use in Python** on the website (Section 1.15).

PROJECT, EXPERIMENT Or leave the code empty and type the project and experiment names exactly as they appear on the website.

When the cell runs, it prints `Using:` and the experiment's name.

No project yet? The optional cell below it creates one: type a name in `NEW_PROJECT`, choose an `ARENA`, and run that cell. Then go back to step 3, type the new project's name in `PROJECT`, and type the project's name followed by `baseline` in `EXPERIMENT`.

2.6 Step 4: look at and change the setup

4. Look at the setup

```
print("Seed:      ", exp.seed)
print("Doors:      ", exp.doors)
print("Stimuli:    ", exp.stimuli)
print("Treatment:", exp.treatment)
```

```
Seed:      1009
Doors:     {Door('north_gate', open), Door('west_gate', open), Door('south_gate', open), Door('dark_gate', open), Door('for
age_gate', open), Door('bridge_entry_gate', open), Door('bridge_exit_gate', open)}
Stimuli:    {Stimulus('bridge_light', light, on, intensity=500), Stimulus('dark_odor', odour, off, intensity=0.7), Stimulus
('foraging_tone', sound, off, intensity=0.4), Stimulus('center_light', light, on, intensity=350)}
Treatment: Treatment(condition='medicine', dose=10.0, unit='mg', administration_step=50.0)
```

Change the setup (optional)

Remove the `#` in front of a line and change it. Use the names printed above. Changes are saved to the website.

```
# exp.seed = 42
# exp.doors["north_gate"].open = False
# exp.stimuli["center_light"].active = True
# exp.treatment.condition = "placebo"

if exp.has_unsaved_changes:
    exp.save()
    print("Saved.")
```

Step 4 prints the experiment's seed, doors, stimuli and treatment. The words in quotes, such as 'north_gate' or 'center_light', are the **names** you use to change something.

To change the setup, in the cell below it:

1. Delete the `#` (and the space after it) at the start of a line.
2. Change the name or value, keeping the quotation marks around names.
3. Run the cell. It saves your changes to the website and prints **Saved**.

```
exp.seed = 42                # a different seed gives a different run
exp.doors["north_gate"].open = False    # close this door at the start
exp.stimuli["center_light"].active = True    # switch this light on at the start
exp.treatment.condition = "placebo"        # no_treatment, placebo or medicine
```

Tip

True means on or open; **False** means off or closed. Both must start with a capital letter.

Your project role decides what you may change. If you change something outside your role, saving is refused and the message lists what you *may* change.

2.7 Step 5: run

5. Run

5000 steps is about 8 minutes in the simulation. The first run takes a minute to download the simulator.

```
STEPS = 5000 #@param {type:"integer"}
```

```
run = exp.run(steps=STEPS)
print("Done:", run)
```

```
Done: RunResult('Anxiety Study baseline', 5000 steps, seed 1009, most time: center 160.8s, corridor_n 156.7s, corridor_s 10
0.5s)
```

STEPS is how long the run lasts: 5000 steps is 500 seconds of simulated time and takes only a few seconds to compute. The cell prints a one-line summary, including where the rat spent most time.

Tip

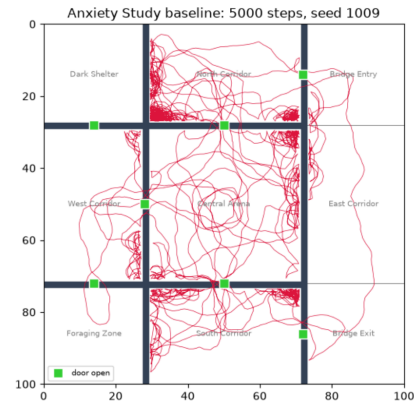
Running the same experiment again with the same seed gives exactly the same result, on any computer. To see natural variation, change the seed.

2.8 Step 6: results

6. Results

The rat's path, then the time it spent in each part of the arena.

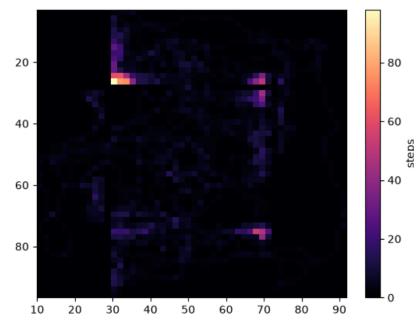
```
import matplotlib.pyplot as plt
run.plot()
plt.show()
run.region_table().round(1)
```



region_id	time_s	entries	first_entry_s
center	160.8	13	0.0
corridor_n	156.7	6	103.1
corridor_s	100.5	6	51.4
corridor_w	42.5	4	18.9
bridge	16.4	3	200.2
bridge_entry	13.7	4	205.2
bridge_exit	6.8	2	197.1
foraging	2.6	1	23.1

Where it spent the most time (brighter = longer):

```
data = run.to_dataframe()
plt.hist2d(data["x"], data["y"], bins=50, cmap="magma")
plt.gca().invert_yaxis()
plt.colorbar(label="steps")
plt.show()
```



Every step as a table:

```
data.head()
```

	step	time	x	y	region_id	active_stimuli	interventions
0	0	0.0	51.527939	49.283615	center	bridge_light;center_light	
1	1	0.1	51.696259	49.215809	center	bridge_light;center_light	
2	2	0.2	52.415516	48.742695	center	bridge_light;center_light	
3	3	0.3	53.656998	48.349625	center	bridge_light;center_light	
4	4	0.4	54.947941	47.846256	center	bridge_light;center_light	

The trajectory plot. The rat's path is drawn in red over the arena. Grey outlines are regions, dark bars are walls, and squares are doors: **green** if open at the end of the run, **black** if closed, with an **orange border** if it was opened or closed during the run.

The region table. One row per region the rat visited:

<code>time_s</code>	Total seconds spent in the region.
<code>entries</code>	How many times the rat entered it.
<code>first_entry_s</code>	When the rat first entered it, in seconds from the start.

The heat map. Where the rat spent the most time: brighter means longer.

The step-by-step table. One row per step:

<code>step, time</code>	The step number, and the time in seconds.
<code>x, y</code>	The rat's position in centimetres.
<code>region_id</code>	The region the rat was in.
<code>active_stimuli</code>	The stimuli that were on at that moment.
<code>interventions</code>	Any change made during the run at that step, such as a door being closed.

2.9 Steps 7 and 8: save to the website and download the data

7. Save to the website

```
run.save()
print("Saved. Open the experiment on the website and go to Review to replay it.")
```

Saved. Open the experiment on the website and go to Review to replay it.

Step 7 saves the run to the website, where it appears under **3 Review** for this experiment and can be replayed in 3D (Section 1.14).

8. Download the data

```
run.save_csv("rodent_run.csv")

from google.colab import files
files.download("rodent_run.csv")
```

Step 8 downloads the step-by-step table as a **CSV file**, which opens in Excel, Google Sheets, R, SPSS or Python. Your browser saves it to your usual **Downloads** folder.

Important

The website replays a run on the experiment's *saved* setup. So if you changed the setup without saving it, the notebook refuses to save that run to the website, instead of showing the rat moving through the wrong walls. Save the setup first (step 4 does this), then run again.

2.10 Doing more

You can add your own cells: click **+ Code** at the top of the notebook, type the lines, and run the cell.

Change doors or stimuli during a run. For example, close the north gate halfway through and switch off the centre light at the same moment:

```
run = exp.run(steps=5000, changes={2500: {"doors": {"north_gate": False},
                                           "stimuli": {"center_light": False}}})
```

These changes are recorded in the data, exactly like changes made on the website.

Compare two conditions. Copy the experiment, change the copy, and run both:

```
dim = exp.save_as_new("Dim centre light")
dim.stimuli["center_light"].intensity = 50
dim.save()
exp.run(steps=5000).region_table()
dim.run(steps=5000).region_table()
```

Earlier runs. `exp.replays()` lists the runs saved for an experiment. To analyse one of them again, download it with `exp.load_replay("id")`.

The full guide to the Python package is at <https://virtual-rodent-docs.pages.dev/>.

2.11 Coming back later

Colab clears its memory when you close the tab or leave it idle for a while. To continue later, open your copy from Google Drive (folder **Colab Notebooks**) and choose **Runtime** → **Run all** again. Your secrets are kept, and everything saved to the website stays there.

2.12 If something goes wrong in Colab

When something is refused, the message explains why and what to do. The most common ones:

Message	What to do
Login failed: check your email and password	Check the two secrets (Section 2.2). Try the same email and password on the website.
Website email: appears asking you to type	A secret is missing, misspelled, or its Notebook access is off. You can also just type them.
No project called ... is visible to you	Check the spelling, including capital letters, or use the experiment code from the website. The message lists your projects.
No experiment called ...	Check the spelling. The message lists the project's experiments.
2 projects are called ...	Several projects share that name. Use the experiment code from Use in Python instead.
Your role ... does not allow some of these changes. You may change: light	Undo the changes that are not in the list, or ask your project lead for more access.
... is published ... Continue from it with save_as_new	Published versions are frozen. Make a copy first: <code>exp = exp.save_as_new("New name")</code> .
Someone saved ... after you loaded it	A colleague changed it meanwhile. Run step 3 again to load their version, then redo your change.
This run used unsaved changes canceling statement due to statement timeout	Save the setup (step 4), then run again. The website was slow to store a long run. Run the save cell again; nothing was half-saved. If it keeps happening, use fewer steps.
NameError: name 'exp' is not defined	The cells above were not run. Use Runtime → Run all .

If the notebook behaves strangely, choose **Runtime** → **Restart session**, then **Run all**.

3 Training your own rat model

Instead of the built-in rat, your own model can control the rat. It practises again and again, gets a reward for what you want it to learn, and gets better. Afterwards you can look at any of its tries, use the trained rat in your experiments, and share it with your project.

The **training notebook** does all of this as a worked example, written the way a researcher would write it up: aim, method, training, results, saving and reuse. In the example the rat starts in the bright room of the light/dark box, next to the lamp, and learns to reach the dark room. Training takes about five to ten minutes.

Open it from **Use in Python** on the website (Section 1.15), or from https://colab.research.google.com/github/HuzaifaMajeed/rodent-sdk-releases/blob/main/notebooks/train_your_rat.ipynb. Save a copy (**File** → **Save a copy in Drive**), make sure your Colab secrets are set (Section 2.2), and choose **Runtime** → **Run all**. You only change the cells whose heading ends with a pencil symbol; everything else runs as it is.

A few words used in this section:

- An **episode** is one try. The rat is placed at its start and your model controls it for a set number of steps (150 steps is 15 seconds), then everything resets for the next try.
- The **observation** is what the rat senses before each step: 31 numbers describing its position and direction, the distance to the nearest wall in eight directions, and how strong the light, sound and odour are, on which side, and where they come from.
- The **action** is the move the model makes: stay, forward, turn left, turn right or run.
- The **reward** is a number the rat gets after every step: higher means better. You decide what it rewards.
- A **round** is a batch of 8 episodes, after which the model learns from them.

3.1 Steps 1 to 3: set up, sign in and choose the experiment

Step 1 installs the RODENT package and downloads the model file, `rat_model.py`, next to the notebook. Steps 2 and 3 are the same as in the starter notebook (Section 2): sign in, then paste the experiment code from **Use in Python**. If you have no light/dark box experiment yet, type a name into `NEW_PROJECT` and the cell creates a project from the Light/Dark Box template.

Changing the setup (optional). The next cell switches lights, odours and sounds on or off, adds a light, or opens a door. Remove the `#` in front of the lines you need. A source works only if it is *enabled* (in the arena) and *active* (on at the start); the templates ship their odour and sound sources disabled. The cell checks the setup and saves it, because training always uses the experiment as it is saved on the website.

```

print(exp.stimuli)

# Switch on a source the arena already has (use a name printed above):
# exp.stimuli["light_source"].enabled = True
# exp.stimuli["light_source"].active = True
# exp.stimuli["light_source"].radius = 35          # how far it reaches, in cm
# exp.stimuli["light_source"].intensity = 700      # brightness (lux) or strength

# Add a new light (the same fields the website uses). Every source needs its own id; the website
# names its lights light_1, light_2, ..., so use a different name. Running this twice adds it once.
# new_light = {
#     "id": "python_light_1", "type": "light", "position": [75.0, 20.0], "radius": 30.0,
#     "initially_active": True, "enabled": True, "object_type": "light", "rotation_y": 0.0,
#     "placement_mode": "floor",
#     "properties": {"lux": 500, "label": "Extra light", "light_kind": "omni", "color": "ffffa8",
#                   "range": 30, "cone_angle": 179},
# }
# if new_light["id"] not in exp.stimuli:
#     exp.config["stimuli"].append(new_light)

# Open or close a door:
# exp.doors["dark_entry"].open = True

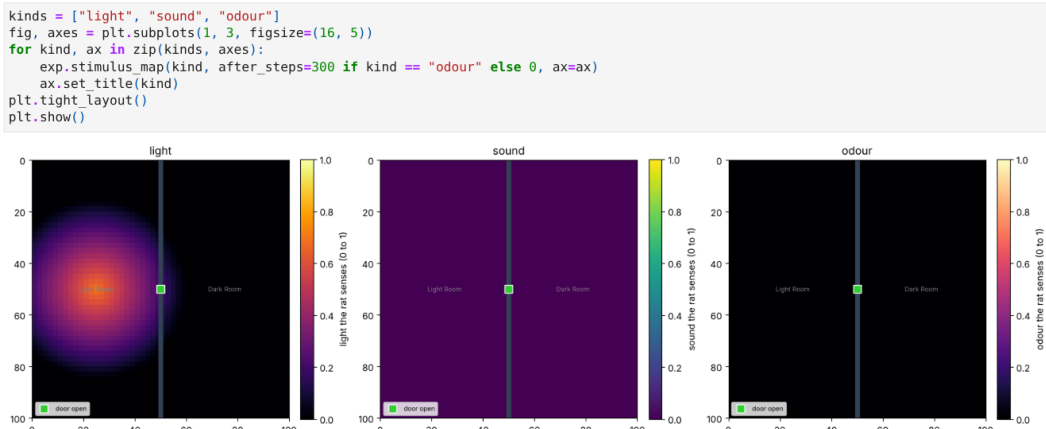
if exp.has_unsaved_changes:
    problems = exp.validate()
    if problems:
        raise ValueError(f"The simulator would reject this setup: {problems}. Nothing was saved; "
                        f"run step 3 again to start from the saved experiment.")
    exp.save()
    print("Saved:", exp.stimuli)

{Stimulus('light_source', light, on, intensity=700), Stimulus('dark_odor', odour, off, intensity=0.7)}

```

3.2 Step 4: explore the experiment

The notebook prints the regions, doors and stimuli (on, off or not enabled), then draws what the rat could sense across the arena: brighter means stronger. Light travels in straight lines and stops at walls and closed doors, so the dark room stays dark. Sound and odour weaken through walls; odour is a gas, so its map is taken after 300 steps. Here only the lamp is on.



It also runs the built-in rat once, as the baseline your trained rat is compared with later.

3.3 Step 5: define the task

TASK is what you want the rat to learn. Each task is a reward and a measure of how well it worked:

5. Define the task

TASK is what you want the rat to learn. Each task is a reward (scored after every step) and a measure (how you judge whether it worked):

TASK	What the rat should learn	Reward after each step	Measure
<code>reach_region</code>	get to the region <code>GOAL</code>	+1 inside <code>GOAL</code> ; before that, a small minus that shrinks as it gets closer	% of episodes that end in <code>GOAL</code>
<code>avoid_light</code>	stay out of the light	minus how bright it is (0 in the dark)	% of the time spent in the dark
<code>follow_odour</code>	find the odour source	how strong the odour is	odour strength over the last 20% of each episode
<code>follow_sound</code>	find the sound source	how loud the sound is	loudness over the last 20% of each episode

Tips for the other tasks

- `follow_odour` and `follow_sound` need a source of that kind that is **enabled** and **active**. On the website, select the source in the arena editor, tick **Enabled** and **Active at start**, and save. In the light/dark box, the odour source sits in the dark room.
- Odour spreads slowly, so give it longer episodes (`STEPS_PER_EPISODE = 300`).
- Sound is only heard inside its radius (18 cm in the templates). Make the radius larger on the website (50 cm works) or start the rat near it.

The other settings

Setting	What it does
<code>GOAL</code>	the region for <code>reach_region</code>
<code>START</code>	where the rat starts each episode, <code>x</code> , <code>y</code> in cm (facing a random way). Empty = a random spot each time. 20, 50 is next to the lamp in the light/dark box
<code>STEPS_PER_EPISODE</code>	length of each episode (150 steps = 15 seconds)
<code>ROUNDS</code>	how long to train. Each round is <code>EPISODES_PER_ROUND</code> episodes followed by one learning step
<code>SEED</code>	makes the training repeatable: the same seed gives the same result

- `reach_region`: get to the region `GOAL` (the example uses `dark_room`).
- `avoid_light`: stay out of the light.
- `follow_odour` and `follow_sound`: find the source. These need an odour or sound source that is enabled and active. Odour spreads slowly, so use longer episodes (`STEPS_PER_EPISODE = 300`); sound is only heard inside its radius, so make the radius larger (50 cm works).

You can also write your own reward instead: it is a small function that receives what just happened (`step.region`, `step.position`, `step.hit_wall`, `step.light`, `step.sound`, `step.odour`) and returns a number. If the rat only gets a reward at the goal, it may never find it; reward it a little for getting closer too.

3.4 Step 6: the model

The model is kept in its own file, `rat_model.py`, which you can open from the folder icon on the left. It is a PPO actor-critic: one small neural network (the *actor*) scores the five moves, and the rat picks moves in proportion to those scores, so it keeps exploring while it learns; a second network (the *critic*) estimates how much reward is still to come. After each round, moves that went better than the critic expected become more likely.

To use your own model, upload your own `rat_model.py` with the same three parts: `forward` (31 numbers in, a score for each move out), `act` (choose a move) and `learn` (improve from finished episodes). The file explains each one.

3.5 Step 7: train

First the untrained model is tested on 20 fixed episodes. Then training runs round by round and prints progress every 5 rounds. Each training run keeps every episode in its own log folder, so none is lost if Colab disconnects. If the measure is still rising at the end, run the training cell again: it carries on from where it stopped.

```

for _ in range(ROUNDS):
    episodes, measures = [], []
    for _ in range(EPISODES_PER_ROUND):
        episode, infos = play(env)
        episodes.append(episode)
        measures.append(measure(infos))
    losses = model.learn(episodes)
    history.append({"round": len(history) + 1,
                  "mean_reward": sum(sum(e["rewards"]) for e in episodes) / len(episodes),
                  "measure": sum(measures) / len(measures), **losses})
    if len(history) % 5 == 0:
        row = history[-1]
        print(f"round {row['round']:>3}: mean reward {row['mean_reward']:.7.1f} | {MEASURE_NAME}: {row['measure']:.2f}")

history_table = pd.DataFrame(history)

round  5: mean reward   -56.1 | % of episodes ending in dark_room: 25.00
round 10: mean reward   -12.5 | % of episodes ending in dark_room: 50.00
round 15: mean reward    -7.7 | % of episodes ending in dark_room: 50.00
round 20: mean reward  -19.7 | % of episodes ending in dark_room: 37.50
round 25: mean reward   14.2 | % of episodes ending in dark_room: 62.50
round 30: mean reward   80.9 | % of episodes ending in dark_room: 100.00
round 35: mean reward   73.2 | % of episodes ending in dark_room: 100.00
round 40: mean reward   84.8 | % of episodes ending in dark_room: 100.00
round 45: mean reward   55.3 | % of episodes ending in dark_room: 75.00
round 50: mean reward   98.3 | % of episodes ending in dark_room: 100.00
round 55: mean reward   71.0 | % of episodes ending in dark_room: 87.50
round 60: mean reward   90.5 | % of episodes ending in dark_room: 87.50
round 65: mean reward   75.6 | % of episodes ending in dark_room: 87.50
round 70: mean reward  106.7 | % of episodes ending in dark_room: 100.00
round 75: mean reward  118.1 | % of episodes ending in dark_room: 100.00
round 80: mean reward  117.2 | % of episodes ending in dark_room: 100.00

```

3.6 Step 8: results

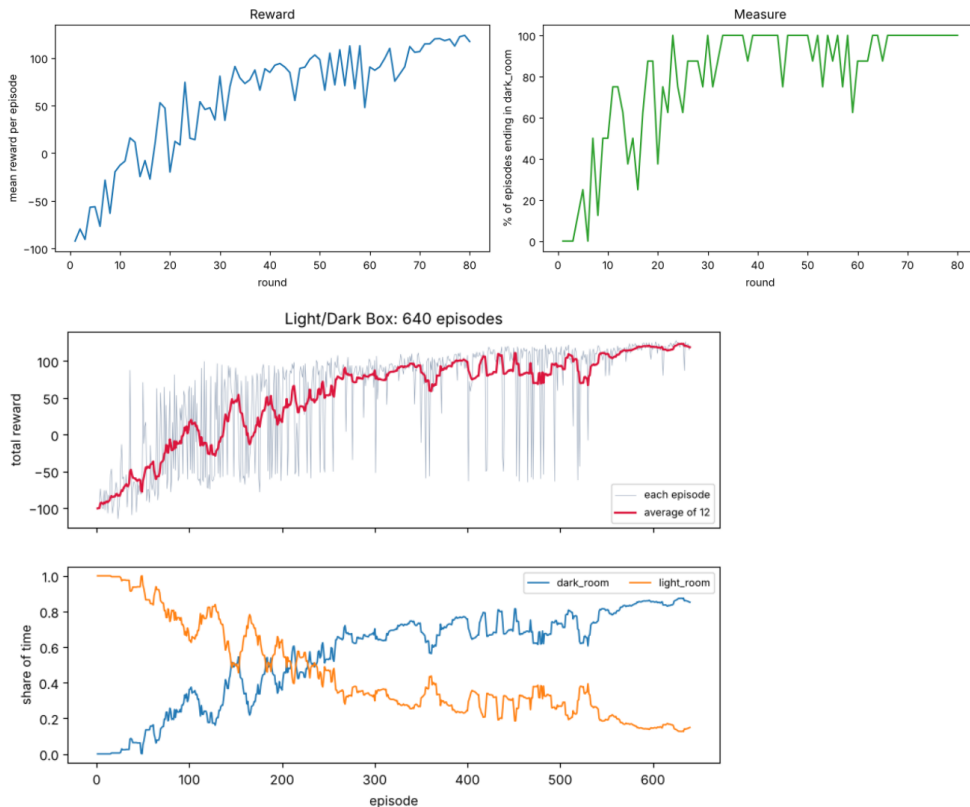
Learning curves. The reward and the task's measure for every round (top), and RODENT's own view of every episode: its total reward, and how its time was split between the rooms (bottom).

```

fig, (left, right) = plt.subplots(1, 2, figsize=(13, 4))
left.plot(history_table["round"], history_table["mean_reward"])
left.set(xlabel="round", ylabel="mean reward per episode", title="Reward")
right.plot(history_table["round"], history_table["measure"], color="tab:green")
right.set(xlabel="round", ylabel=MEASURE_NAME, title="Measure")
plt.tight_layout()
plt.show()

# RODENT's own view of the log: reward per episode, and how each episode's time was split between regions.
env.log.plot()
plt.show()

```



Before and after. The same 20 test episodes as before training, and a short summary. In this example the rat reached the dark room in none of them before training and in all 20 after.

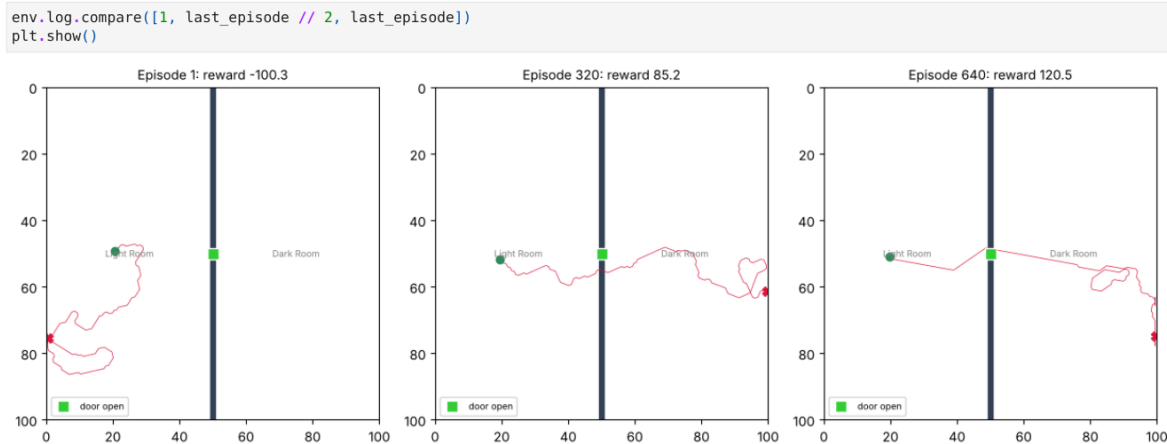
```
after = test()
print(f"{MEASURE_NAME}")
print(f" before training: {before:.2f}")
print(f" after training: {after:.2f}")

% of episodes ending in dark_room
before training: 0.00
after training: 100.00

print(f"Task: {TASKS[TASK]['description']} ({exp.name})")
print(f"Training: {len(history)} rounds, {len(history) * EPISODES_PER_ROUND} episodes of {STEPS_PER_EPISODE} steps")
print(f"{MEASURE_NAME}: {before:.2f} before training, {after:.2f} after")
print(f"Learning curve: {history[0]['measure']:.2f} in round 1, {history_table['measure'].tail(5).mean():.2f} over the last 5 rounds")

Task: Reach the region dark_room (Light/Dark Box)
Training: 80 rounds, 640 episodes of 150 steps
% of episodes ending in dark_room: 0.00 before training, 100.00 after
Learning curve: 0.00 in round 1, 100.00 over the last 5 rounds
```

Look at any episode. Type any episode number into EPISODE (0 means the last one) and a step into AT_STEP. The episode is rebuilt exactly from the log, without the model: you get its path, what the rat sensed at that step, and a short animation. The next cell shows the first, a middle and the last episode side by side:



The trained rat next to the built-in rat. The trained model now drives the rat in a normal run of the experiment, with the same tables and plots as any run.

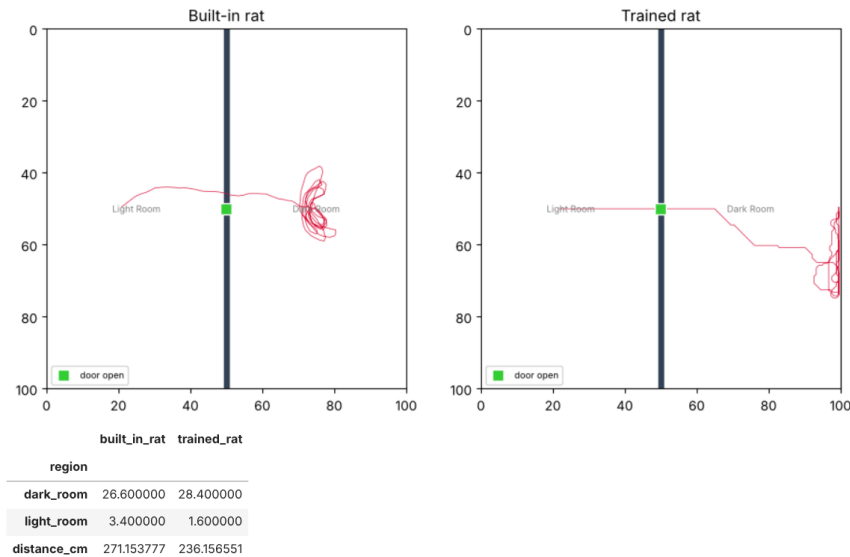
```

model_run = exp.run(steps=2 * STEPS_PER_EPISODE, model=model, actions="discrete",
                    start=start.get("position"), episode_length=STEPS_PER_EPISODE)

fig, (left, right) = plt.subplots(1, 2, figsize=(11, 5))
builtin_run.plot(ax=left)
left.set_title("Built-in rat")
model_run.plot(ax=right)
right.set_title("Trained rat")
plt.show()

rodent.compare(built_in_rat=builtin_run, trained_rat=model_run)

```



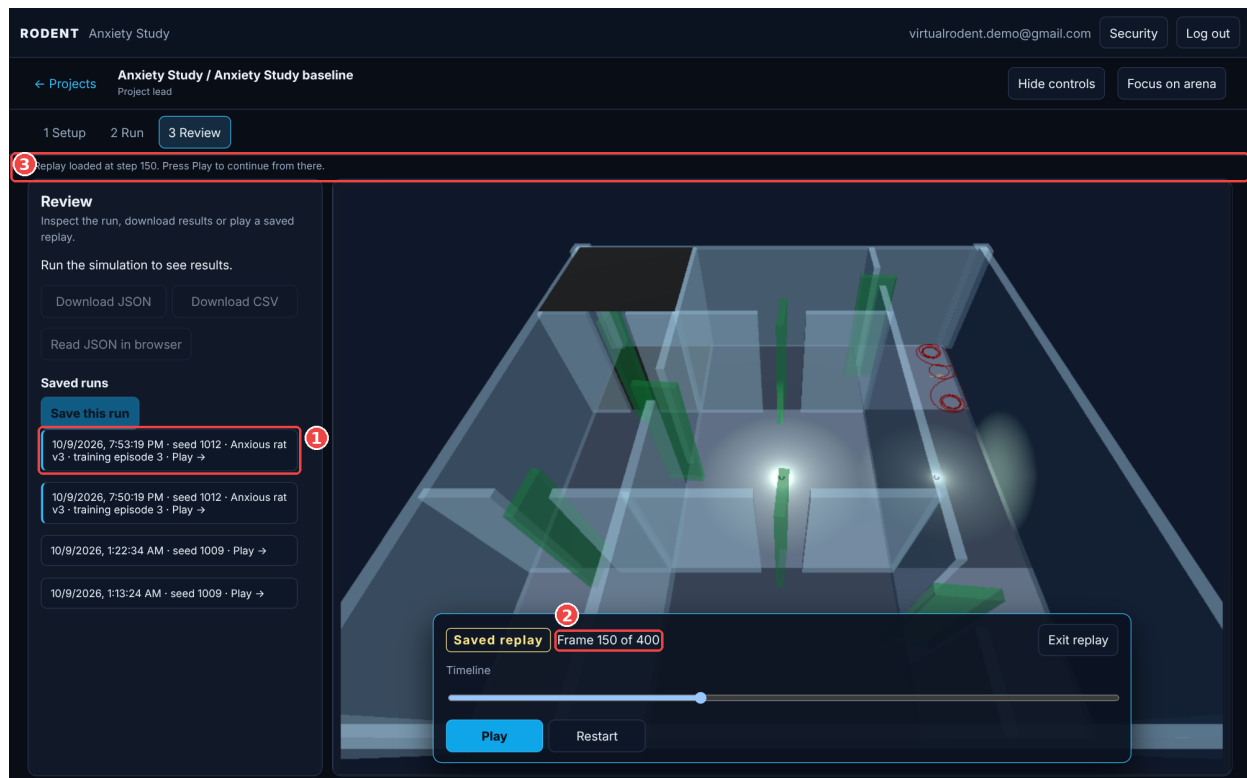
3.7 Step 9: save to your project and find it on the website

Type a name into `MODEL_NAME`. The cell saves the model to your project as a new version (saving again with the same name adds version 2, 3, ...), together with how it was trained, and saves the trained rat's run and the episode you looked at to **Review**. It prints links that open them on the website, the episode at the step you chose.

Saved models are listed on the project page, under **Trained rat models**:

- 1 The model's name. Below it: the latest version, how many episodes it was trained for, on which experiment, and its notes.
- 2 The learning curve of the latest version: its average reward over training.
- 3 **Copy Python to load it:** copies the line that loads the model in Python, so a teammate can run it or keep training it.

Runs driven by a model are marked in **Review**, and a link from Python opens one at a chosen step:



- 1 A saved run driven by a model, with the model's name and version and, for a training episode, its number.
- 2 The replay starts at the step from the link (here step 150 of 400). Press **Play** to continue from there.
- 3 The status line confirms where the replay was opened.

3.8 Step 10: use the model in another experiment

A saved model belongs to the **project**, so every experiment in the project can use it: for example a copy of this experiment with the door closed or a brighter light. Type the other experiment's name into `OTHER_EXPERIMENT` and run the cell. It loads the saved model, runs it in that experiment and saves the run to that experiment's **Review**, labelled with the model and version.

The model reacts to what it senses and to where it is, so it works best in experiments with the same arena layout. In a very different arena, treat it as a starting point and train it further there.

Step 11 downloads your work: the model file, its weights, the training history and every logged episode.

3.9 Good to know

- **Read results against the reward.** A model learns what its reward pays for. With `avoid_light`, for example, the trained rat may wait in an unlit corner of the bright room instead of going to the dark room, and still score well.
- **A trained rat is not a real rat.** It shows what this model learned from this reward in this arena. Report the reward, arena and training settings with any result.
- **Speed.** About 1,000 steps a second, plus the time your model takes. The example (640 episodes of 150 steps) takes about five minutes in Colab.
- **Who can do what.** Everyone in the project can see, load and run its models. The lead and editors can save new versions; versions never change once saved, so results always point to exactly the model that produced them.
- **The website replays, Python runs.** The website lists models and replays their runs; training and running models happen in Python.
- **Saved runs need the saved setup.** An episode can be saved to **Review** only while the experiment on the website is still the setup it was trained on.