Gramophone User Guide

October 5, 2018

FEMTONICS

ADVANCED MICROSCOPY

Contents

1 Introduction

The Gramophone system is a single dimension locomotion tracking device for head-restrained mice that has two modes of operation. It can either be used for high-accuracy velocity recording in conjunction with a two-photon microscope, or as a control interface for behavioural training in a virtual linear maze.

2 Item numbers

Spare parts

- Running disk - `F-FVR-D`
- Head-plates (set of 10) - `F-FVR-HPS10`

System

- Bare-bones - `F-FVR-SK`
- Kit for velocity recording - `F-FVR-KIT1`
- Complete system - `F-FVR-KIT2`

3 Compatibility

The system is compatible with Microsoft Windows 7 or newer. To run the computer software, a Python 3.6 interpreter is required.
The system can be used in conjunction with Femtonics SMART microscopes and certain custom built models. To get further information about compatibility with your device, please contact Femtonics Ltd. at info@femtonics.eu.

4 Links

- Python Package Index - https://pypi.org/project/GramophoneTools
- GitHub - https://github.com/Femtonics/GramophoneTools
- Documentation - http://gramophone.femtonics.eu/
- User guide - http://gramophone.femtonics.eu/user_guide.html

5 Contact

If you need further assistance, please contact Femtonics Ltd. via e-mail at info@femtonics.eu or visit our website http://femtonics.eu/contact

6 Cleaning instructions

Disk

Remove the disk (Figure 1a) by unscrewing its mounting screw (Figure 1b) and clean it with warm soapy water. The disk can be sterilized with alcohol-based disinfectants.

System

Disconnect the device from the computer by unplugging the USB cable (Figure 2b). Clean the device with a damp cloth.

7 Hardware overview

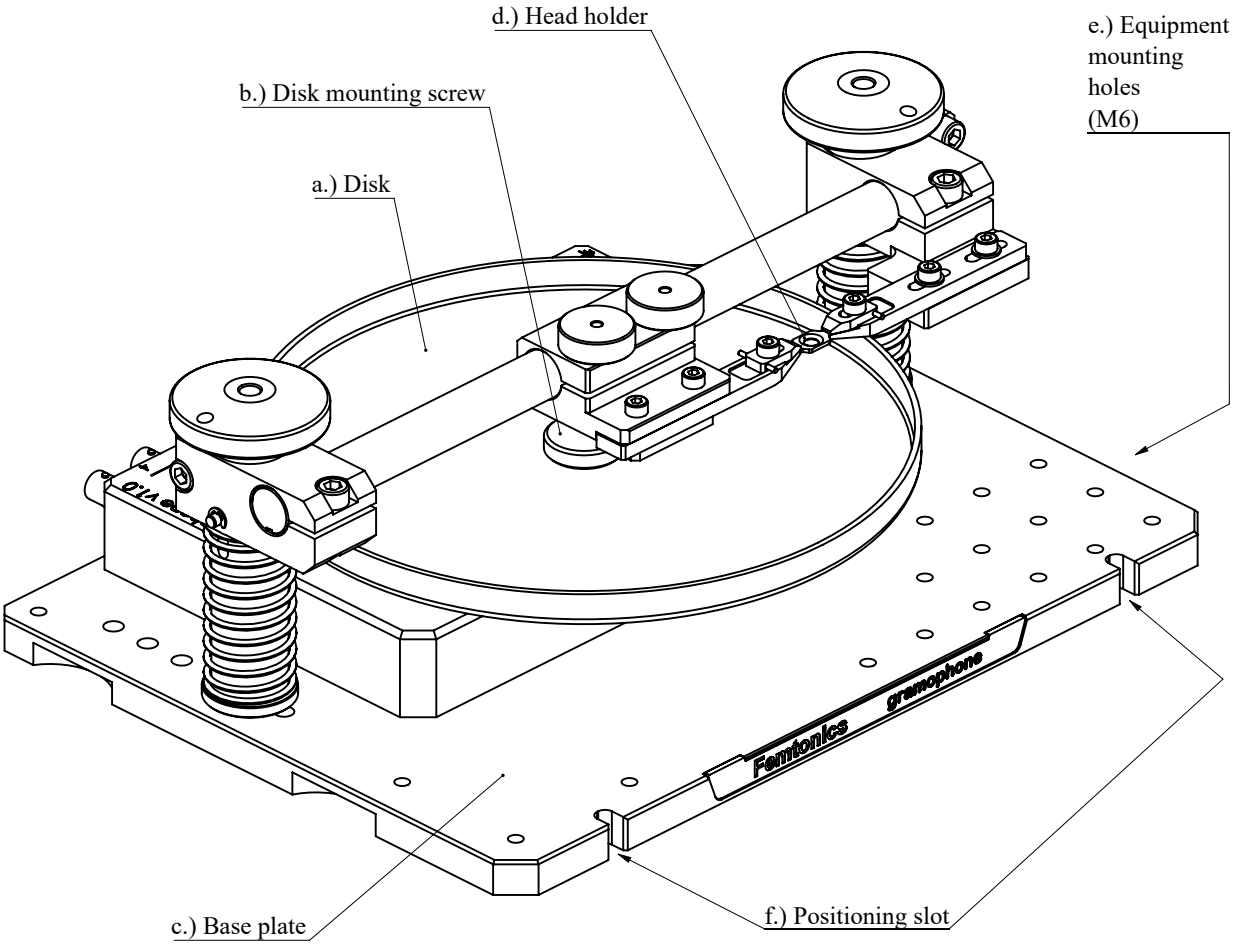
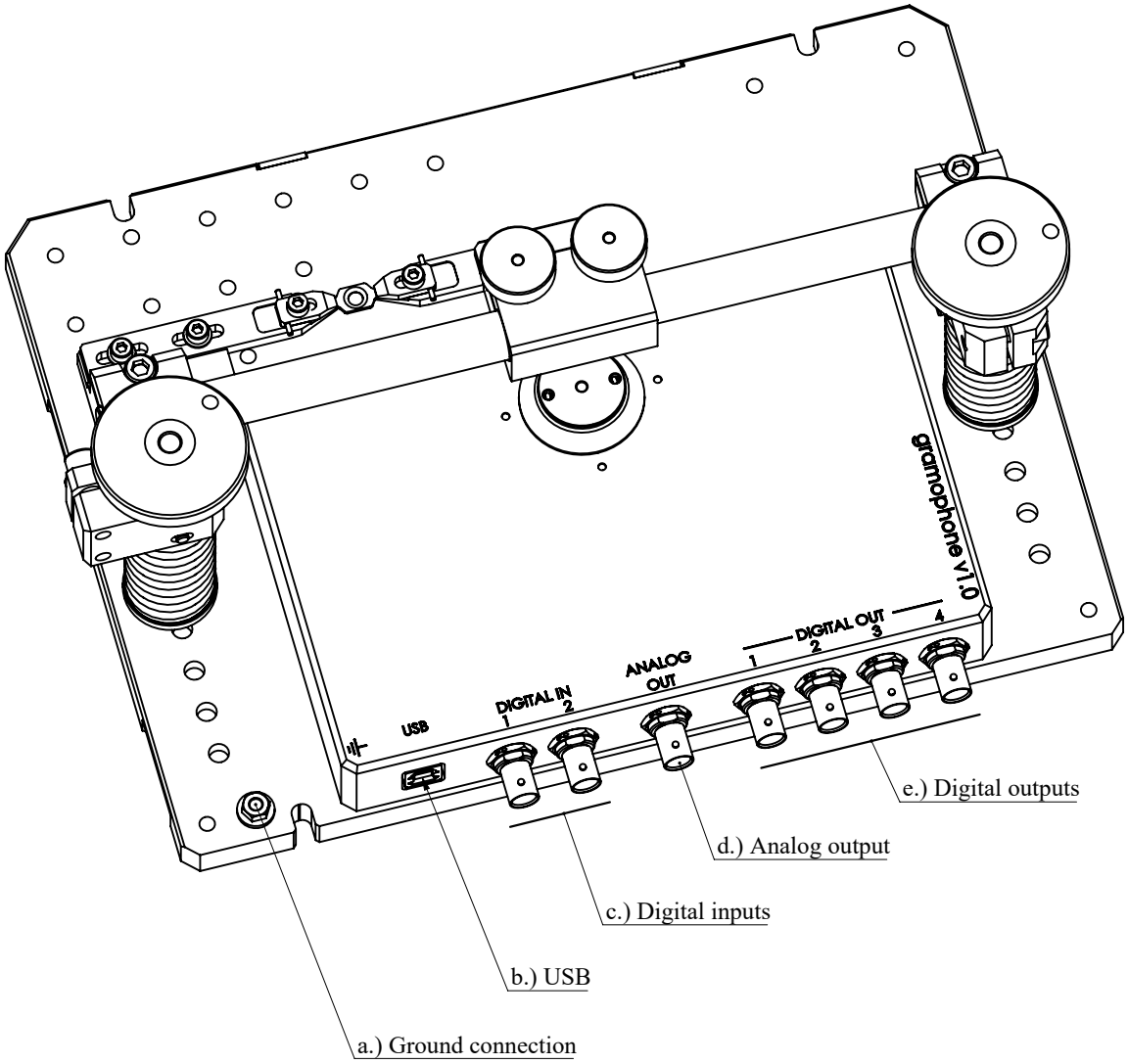


Figure 1: Overview of the Gramophone device.

Specifications

- Operating temperature: 0-60 °C
- Relative humidity during operation: $< 90\%$, avoid condensation
- Power requirements - 5V DC @500mA via USB. The use of passive USB hubs is prohibited.

Inputs and Outputs



;

Figure 2: Inputs and outputs on the Gramophone device.

The Gramophone connects to the computer via USB 2.0. A cable is provided with the system.
The system has 7 full-sized BNC connectors:

- 4 digital outputs (Figure 2e) that are connected to ground if their state is set to low, and connected to +5 volts if their state is set to high.

  high: 4 - 5,5V @ 3mA

  low: 0 - 1V @ -3mA

- 2 digital inputs (Figure 2c) that are considered to be in a low state if the voltage on them is below 1 volts and high if the voltage is above 4 volts.

  high: 4-6V @3mA

  low: 0 - 1V

- 1 analogue differential output (Figure 2d) that is proportional to the current velocity of the animal.

  $V_{\text{common mode}}$: 0V

  $V_{\text{differential}}$: 0 - 2,5V

All digital IOs are galvanically isolated from the control circuit.

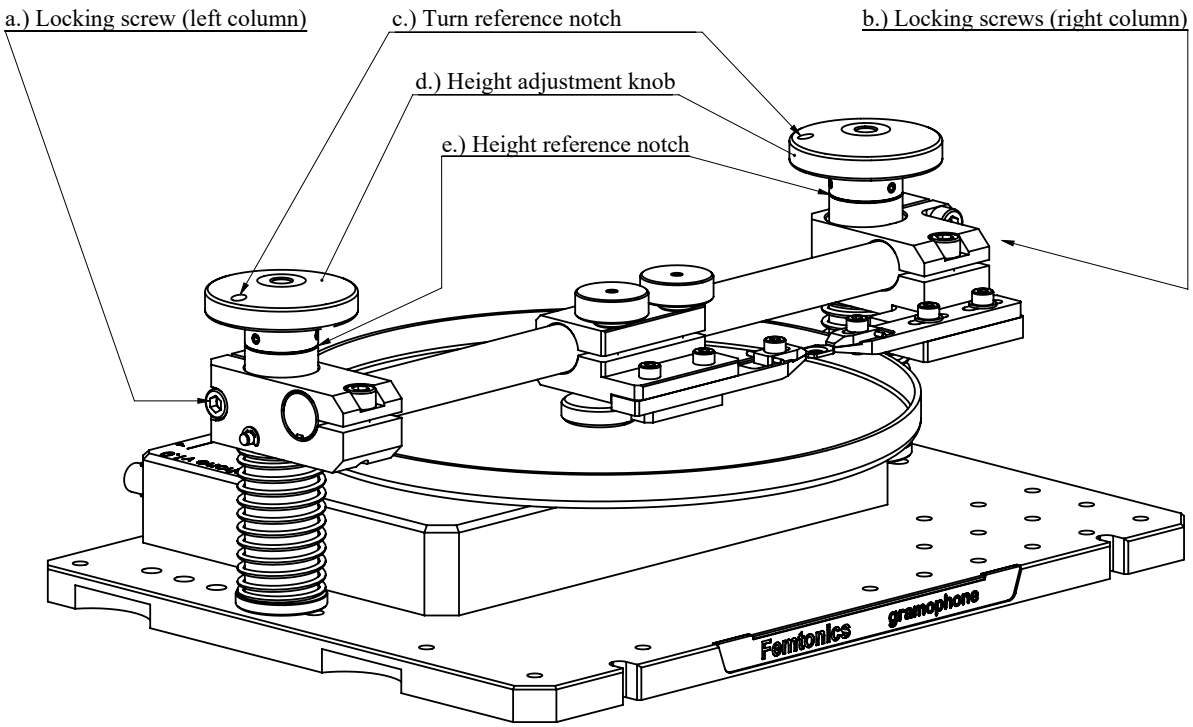
Ground connection

The system has a single banana plug connection (Figure 2a) on the base plate marked with a ground sign. Always connect this to a ground with the included cable before using the device.

Height adjustment

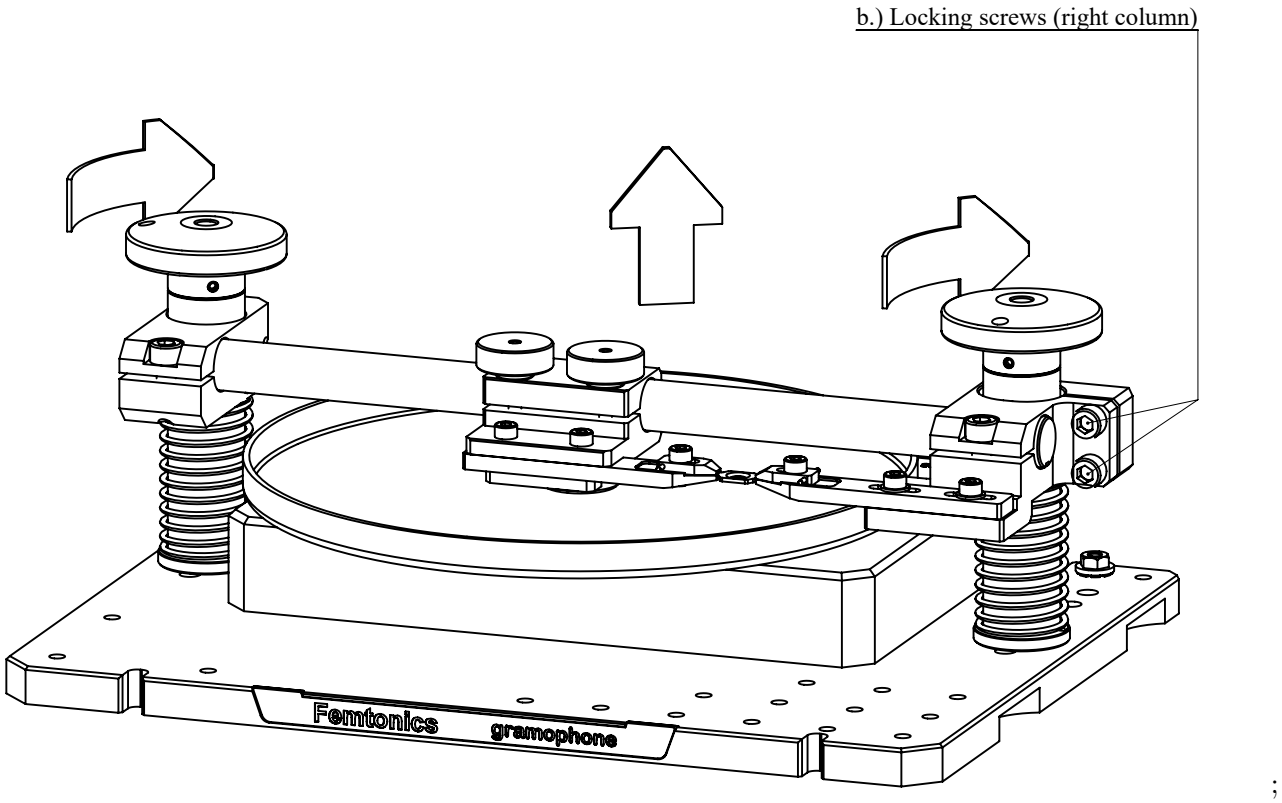
The distance between the head-holder and the disk can be adjusted to fit mice of any size comfortably.

1. Loosen the locking screws with the included hex key (Figure 3a and b).
2. Adjust both height-adjustment knobs (Figure 3d) at the same time until the desired height is reached. Turning the knobs clockwise raises the head-holder relative to the disk, while turning it counter-clockwise lowers it (Figure 4). You can use the height reference notches Figure 3e) to make sure that the head holder is not tilted.
3. Lock the position by tightening the locking screws with the included hex key (Figure 3a and b).



;

Figure 3: Inputs and outputs on the Gramophone device.



;

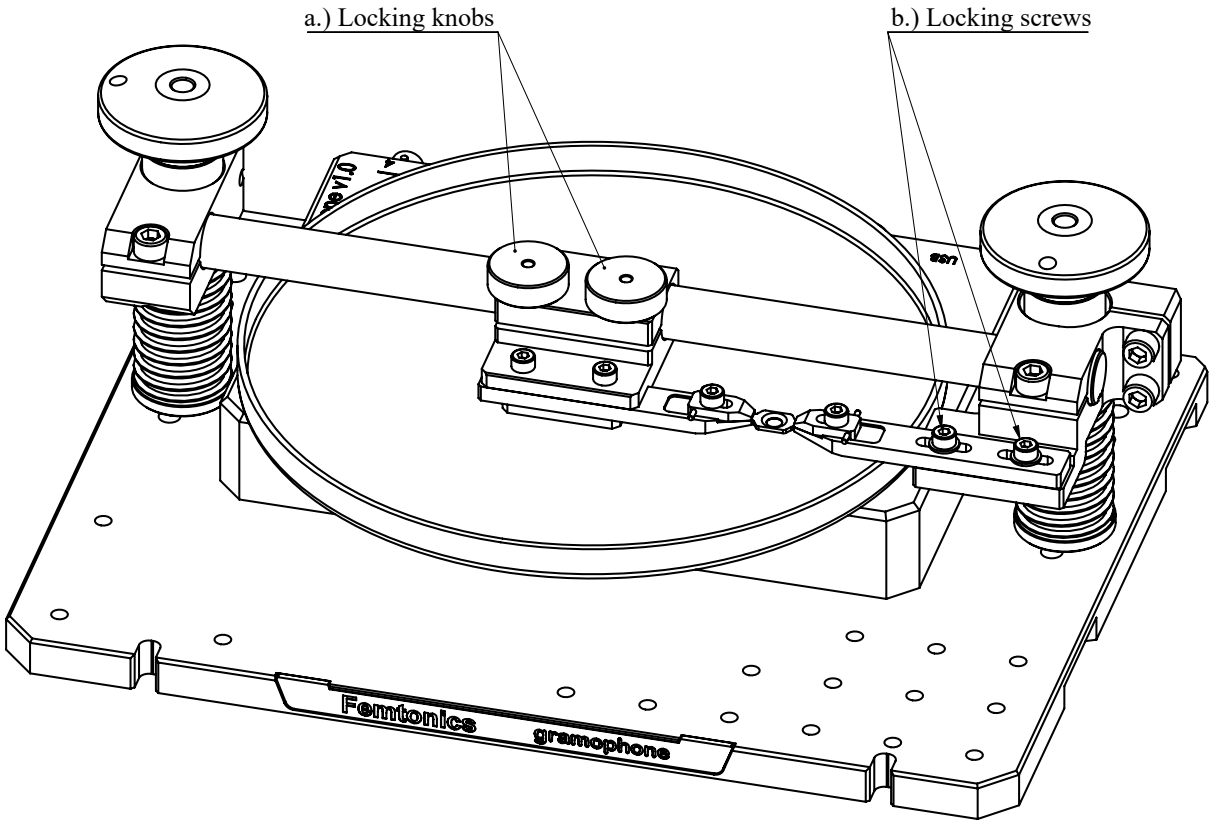
Figure 4: Inputs and outputs on the Gramophone device.

Position adjustment

The position of the head-holder can be adjusted both sideways and back and forth to position the mouse anywhere on the disk.

Sideways adjustment

1. Loosen the two locking screws on the cylindrical axle with the knobs (Figure 5a).
2. Move the left part of the head-holder to the desired position by sliding the whole assembly on the axle.
3. Lock the left part of the head-holder with the knobs in the desired position (Figure 5a).
4. Loosen the two screws (Figure 5b) on the right part of the head-holder.
5. Adjust the right part until the head-holder fits a head-mount perfectly. *Note: It is best practice to keep a head-mount handy for this.*
6. Lock the right part in place with the two locking screws (Figure 5b).

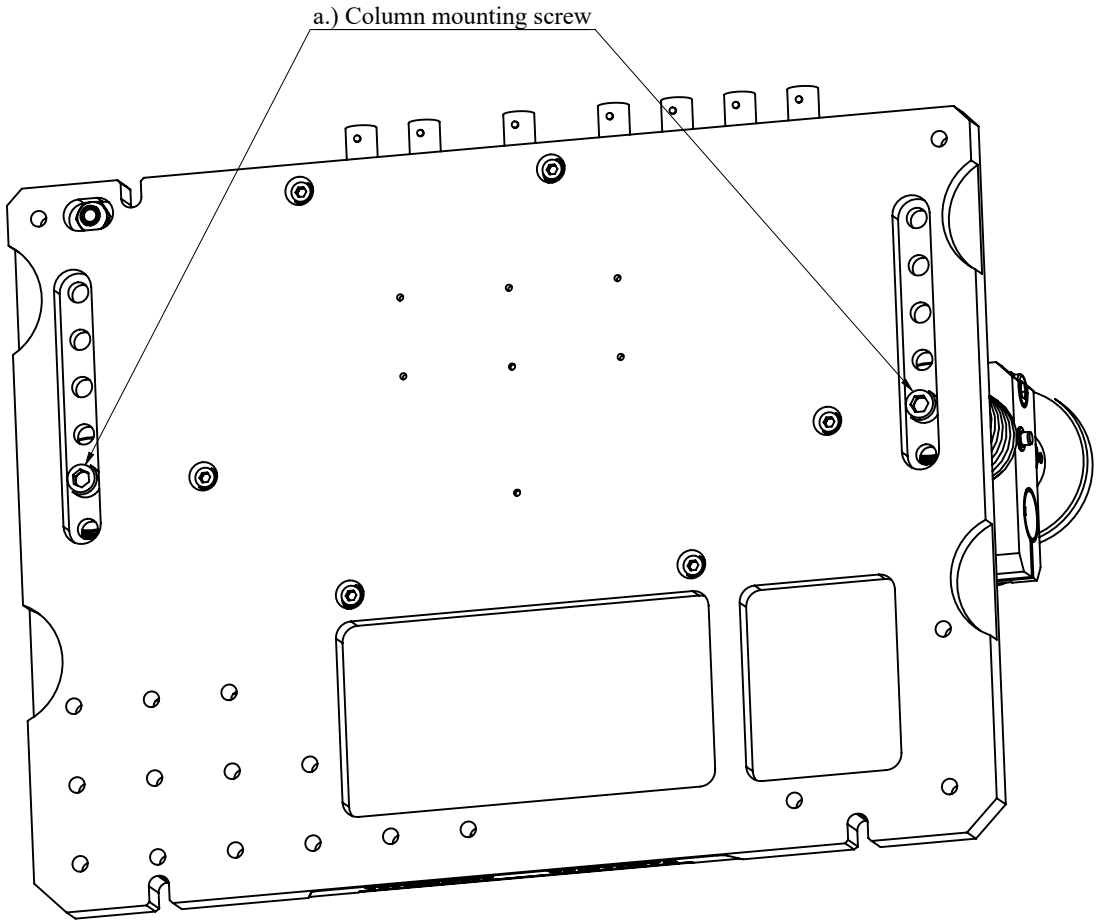


;

Figure 5: Sideways adjustment of the mouse on the Gramophone.

Back-and-forth adjustment

1. Make sure that the height-adjustment screws are tightened (Figure 3a and b).
2. Remove the screws holding the columns in place (Figure 6a) with the included hex key.
3. Position the columns in the desired location such that their centres line up with a pair of mounting holes on the base plate.
4. Put back the removed screws in the new location and tighten them with the included hex key (Figure 6a).



;

Figure 6: Inputs and outputs on the Gramophone device.

8 Software installation

Drivers

No additional drivers are required for the Gramophone system. It operates as a HID (Human Interface Device), for which, all drivers are provided by modern operating systems.

Python interpreter

To run the computer software of the Gramophone system, a Python 3.6 interpreter has to be installed. To install it, download the latest Miniconda installer for your version of Windows (links: **32-bit** or **64-bit**) from the **Conda website**.

Note: If you are not sure which version of Windows you are using, you can check by pressing the Win+Pause/Break key combination to open the System window and check under System → System type, or by right-clicking the Start menu, choosing System and checking under Device specifications → System type.

Follow the installation steps:

1. Start the downloaded .exe file (Figure 7). Click Next.

Figure 7: The Miniconda installer's welcome screen.

2. Agree to the License Agreement (Figure 8).

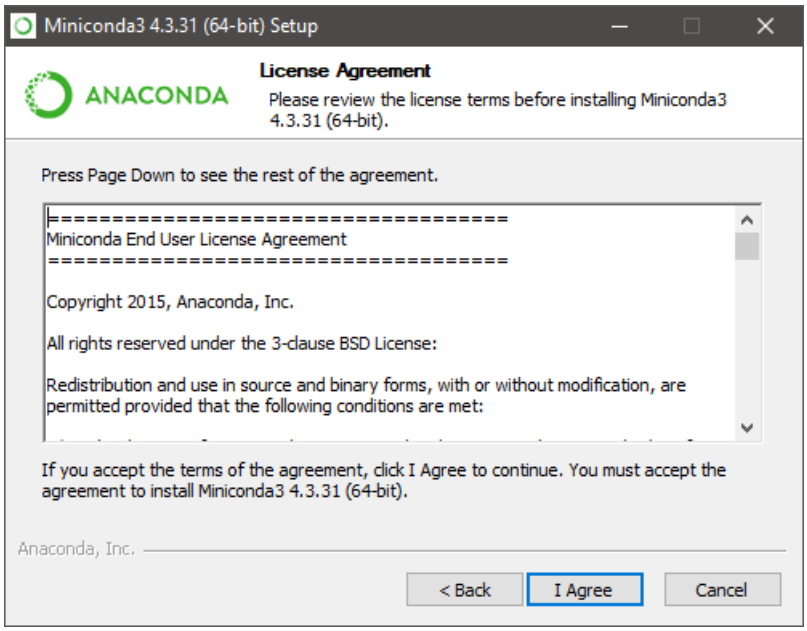


Figure 8: Agree to the License Agreement.

3. If multiple users will be using the same machine, install for all users (Figure 9). Click Next.

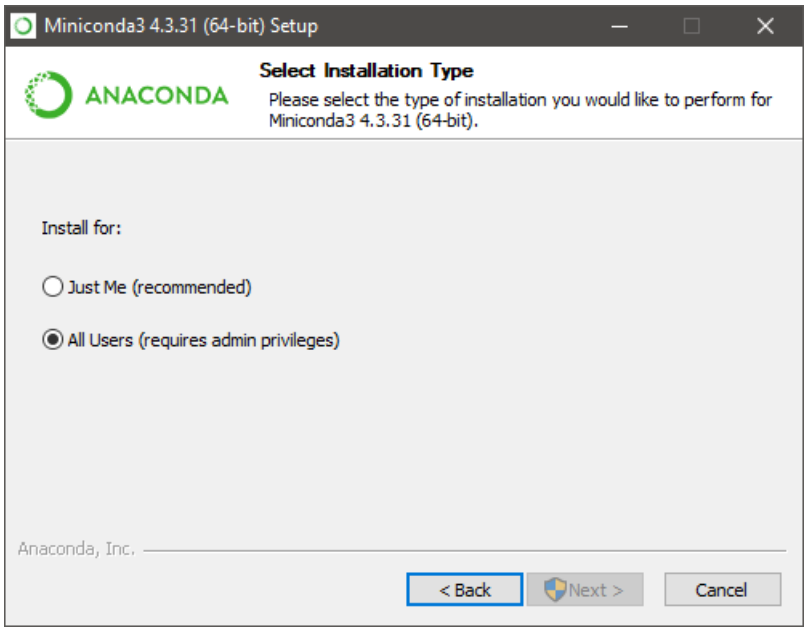


Figure 9: Install for all users if you use multiple users on the computer.

4. Leave the installation directory as the default (Figure 10). Click Next.

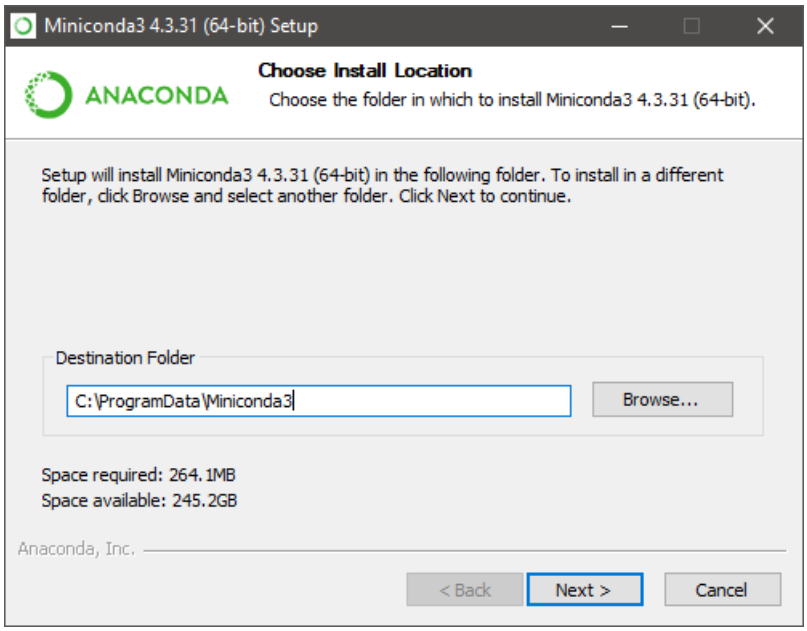


Figure 10: Leave the install directory as the default if possible.

5. Add Anaconda to the system PATH variable and register it as the system's Python 3.2 interpreter (Figure 11). Click Install.

   Note: If you already have other versions of Python installed, leave these options unchecked and make sure you execute all Gramophone software from the Anaconda Prompt.

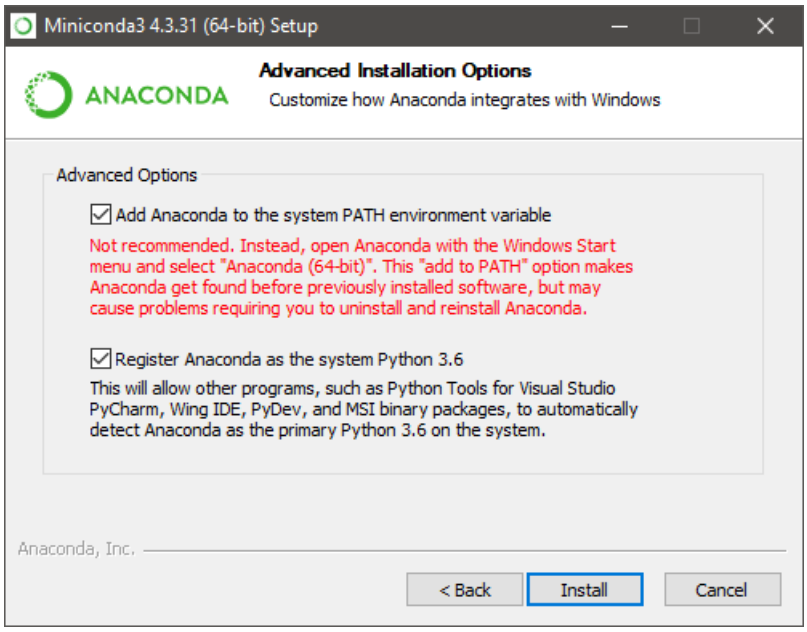


Figure 11: Check both options.

6. Wait for the installation to finish (Figure 12). Click Next.

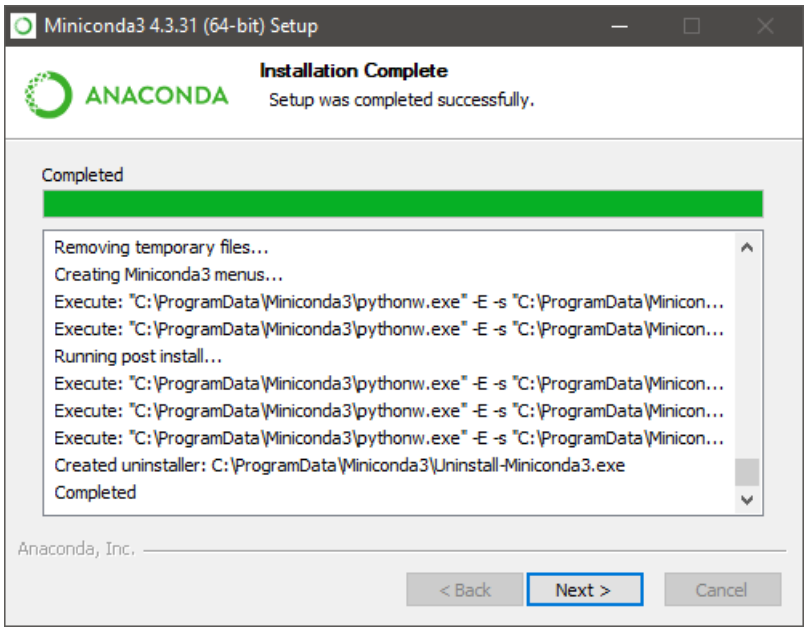


Figure 12: Wait for installation to finish.

7. Exit the installer by clicking Finish (Figure 13).

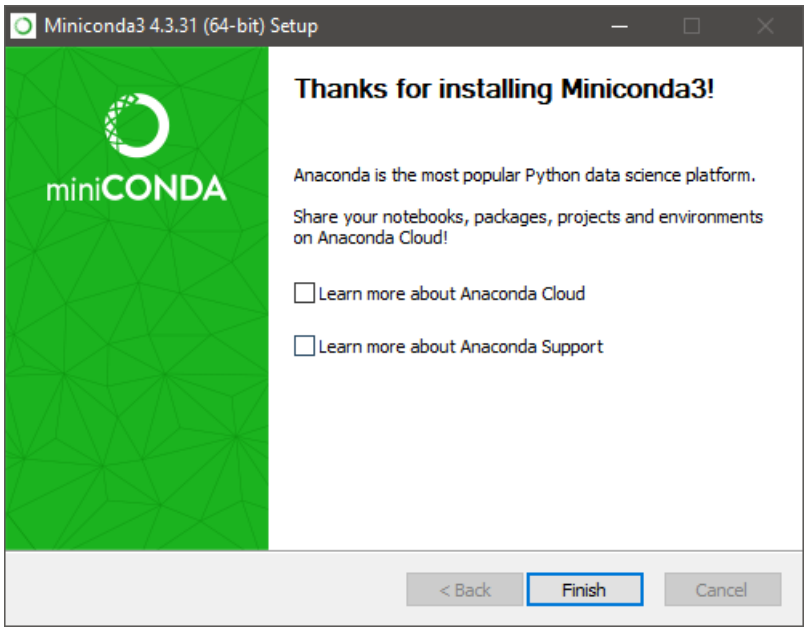


Figure 13: Exit the installer.

Python modules

All Gramophone related modules are collected in a single python package. To install it run the following command **as an administrator** in Windows Power Shell, Command Prompt or Anaconda Prompt:

```
pip install GramophoneTools
```

Note: In Windows 10 you can open an Power Shell window as an administrator by right clicking on the start menu and choosing the "Windows PowerShell (Admin)" option.

Updating

To find out if there is a newer version available run:

```
pip list -o
```

If GramohoneTools shows up in the list you can update it by running:

```
pip install -U GramophoneTools
```

Associating .py files with the Python interpreter

To start Python scripts (files with .py extension) with a double click in Windows you have to associate them with the Python interpreter (python.exe). To do so right click on any file with a .py extension under the "Open with" menu select the "Choose an other app" option. In the popup window choose the "Look for an other app on this PC" (you might need to open the "More apps" section). In the browser window that opened navigate to the location of python.exe. If you installed Miniconda with the default location this should be: `C:\ProgramData\Miniconda3\python.exe`
Note: The ProgramData folder is hidden in Windows so you have to type the location in the address bar to navigate there, or make hidden files and folders visible.

Associating .vlg files with the GramophoneTools Recorder

The velocity recorder produces .vlg files as an output (see: 10). To open these in the Recorder, right click on a .vlg file, select open with, search for an application on the computer and select `gramrec.exe` in the scripts folder of your Python installation. If you installed Python as described above (see: section 8) this can be found here: `C:\ProgramData\Miniconda3\Scripts`.
Note: The ProgramData folder is hidden in Windows so you have to type the location in the address bar to navigate there, or make hidden files and folders visible.
Note: You can also make a shortcut for `gramrec.exe` on the desktop to make starting it easier.

9 Software overview

The GramophoneTools package has three modules. The `Comms` module handles all communication with the device. This can be used to develop your own application for the Gramophone. The Recorder module is an application with a GUI (Graphical User Interface) that can be used to record the animals velocity in a triggered manner. The LinMaze module is a collection of tools used to generate a 2 dimensional linear maze the mouse can navigate. Events and Rules can be added to this maze to implement various conditioning tasks.

After installing the GramophoneTools package, you can use a number of commands. The `gramrec` command simply starts the Recorder application. You can include a path to a .vlg file as a command line argument to open it with the recorder. The `gram` command can be used for various things, run `gram help` to find out more.
Note: You can open a terminal for these commands in Windows 10 by right clicking the start menu and selecting the 'Windows PowerShell' option, or you can run them in the Run window by pressing 'Win+R'.
Note: You can create shortcuts for these commands by right clicking where you want the shortcut to be (eg. on the Desktop) and entering the command as the location of the item.

10 Recorder module

The Gramophone recorder is software that can make high accuracy velocity recordings. To keep the velocity records in sync with two-photon measurements the digital input 1 (Figure 2c) is used as a trigger input for recordings.

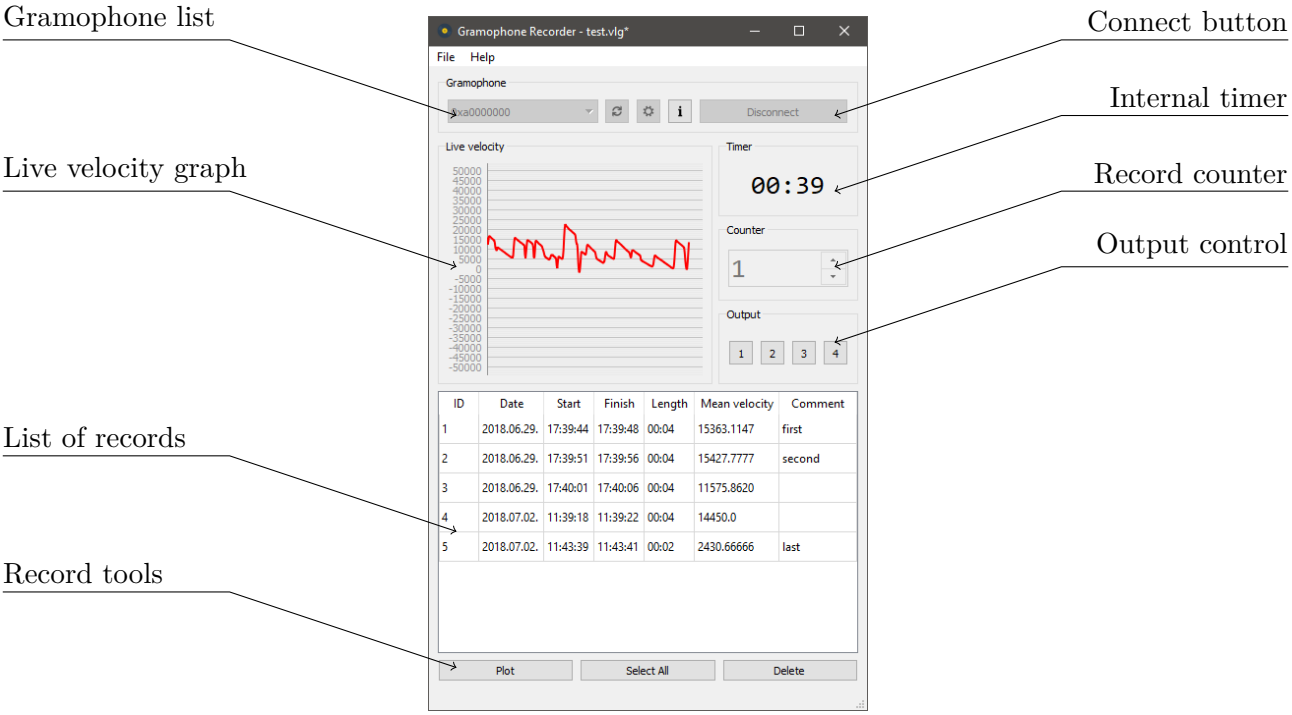


Figure 14: The user interface of the Gramophone recorder during recording.

Usage

To accurately record velocity with the Gramophone system follow these steps.

Hardware setup

1. Fix the mouse in the head holder. If necessary, adjust the height of the holder as explained in the Height adjustment section.

2. Put the Gramophone with the mouse into the microscope, such that the craniotomy window on the head of the mouse lines up with the microscope's objective perfectly.

3. Connect the grounding point to a ground with the included cable (Figure 2a).

4. Connect the digital output of the microscope with a digital input of the Gramophone (Figure 2c) with a BNC cable.

5. Connect the Gramophone to the computer with the included USB cable (Figure 2b).

Microscope configuration

Configure the microscope to pull its digital output to hight for the time period you wish to record (eg.: pull to high when the shutter opens and to low when it closes).

Software setup

1. Start the Gramophone recorder with the icon on the Desktop or from the terminal with the `gramrec` command.

2. Select the Gramophone you wish to record from the dropdown list (see: Figure 14). If you can't find the connected device in the list, you can refresh it with the button next to it.

3. Click the gear button next to the list of Gramophones and in the window that opened set the trigger channel to the input you connected the microscope to.

4. Click the Connect button (see: Figure 14). The current velocity will be plotted on the Live velocity plot.

5. Start your measurement protocol in MES or MESc, the recorder will automatically capture velocity for every measurement unit.

6. You can see the velocity records on the List of records (Figure 14). You can select any number of records with the mouse and plot or delete them with the Record tools below the list.

Recording results

The velocity log file is an HDF5 file with a `.vlg` extension. For more information about it's structure please visit the **online documentation of the Recorder module**.

11 LinMaze module

LinMaze is a simple virtual linear maze system. It can generate patterns based on parameters and stitch these patterns together into an endless loop the mouse can navigate in using the Gramophone. Optionally, you can define rules to condition the mouse to a task eg. to differentiate between different patterns and behave accordingly.

Levels

To use the software you have to make a Python script that defines your Level. A Level is a collection of patterns, events and rules.

To make a new Level you have to make a small Pyton script. All levels should start by importing the Level object and an `if` statement that is needed for multithreading to work properly. After the `if` statement you can start constructing your Level. Be sure to put 4 spaces in front of every line from now on.
Note: You can comment out lines by adding a # character in front or make multiple line comment by wrapping with triple ' characters ''' like this '''. Comments will be coloured green in the following examples.

First create a Level object. In these examples I will call mine `LVL`.

```
''' An example level for LinMaze '''
from GramophoneTools.LinMaze.Level import Level
if __name__ == '__main__':
   LVL = Level(
      name = 'myFirstLevel',
      screen_res = (1280, 1024),
      zone_offset = 720,
      transition_width = 100,
      rgb = (0,0,1)
      )
```

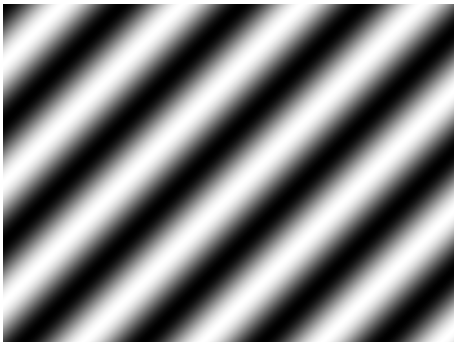
The parameters for the Level are:

- `name` - can be any string, this will show up in default file names.
- `screen_res` - two integer numbers in parentheses, separated by commas. These give the resolution of the screen that the level will be running on.
- `zone_offset` - an integer number that gives where the borders of the zones should be on the screen. Eg. If you set it to half the horizontal resolution, zones will be entered when they reach the middle of the screen from the right and are left when the last pixel of the frame leaves the middle.
- `transition_width` - an even integer that describes how wide the smooth transition should be between the frames. Setting this to 100 for example would make all frames 100 pixels bigger on both ends and in this 100 pixel wide space frames would gradually turn into one an other.
- `rgb` - tuple of 3 floats that give the ratio of colours (red, green and blue in order) eg. (1,1,1) is black and white, (0,0.5,0.8) is black and dark cyan.

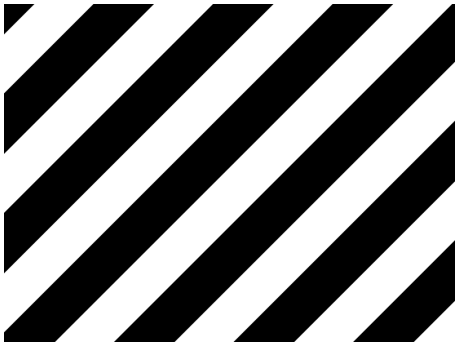
Frames

Frames are rectangular pictures that can be displayed by the system. There are 8 different types of generated Frames you can add to your level (see: Figure 15). In addition you can turn image files into Frames. Frames are always as high as the screen, and as long as the given length parameter. Blocks are Frames with a `zone_type` parameter. This way you can refer to a group of patterns by making zone rules (see: section 11).

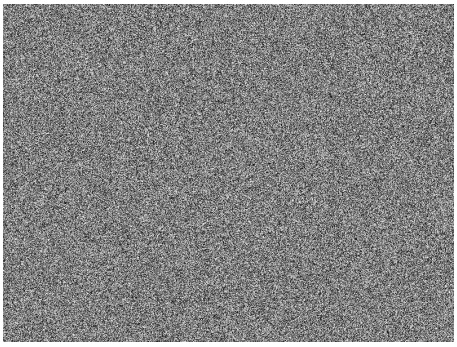
- `length` - An integer number that gives the length of the frame in pixels.
- `random_seed` - For randomized frames (eg. `'cloud'`, `'greynoise'` etc.). Can be any number. If the random seed of two frames of similar type and length is the same, they will be exact copies of



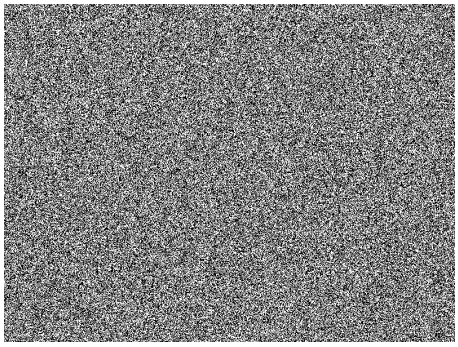
(a) Sine wave modulated Frame



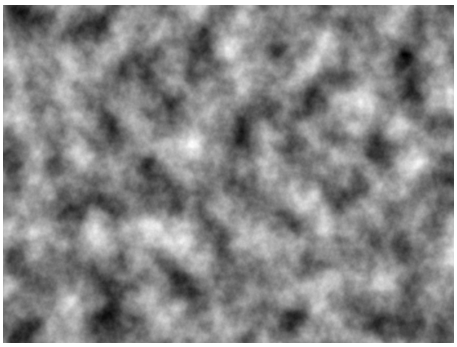
(b) Square wave modulated Frame



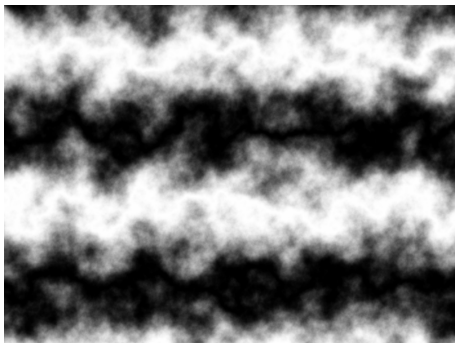
(c) Greyscale noise Frame



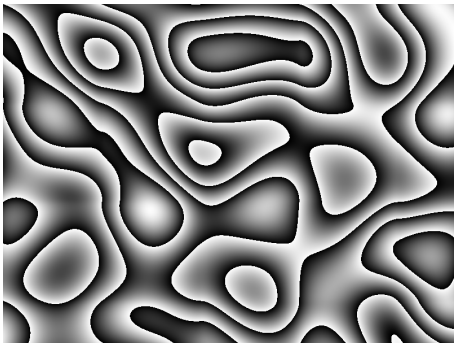
(d) Binary noise Frame



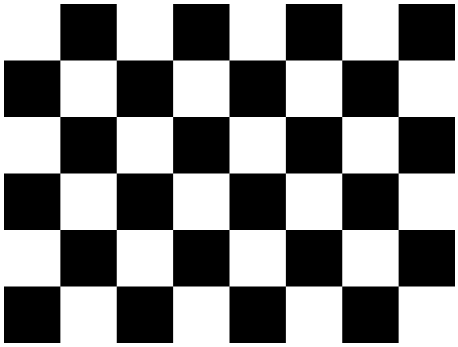
(e) Cloud Frame



(f) Marble Frame



(g) Wood grain Frame



(h) Checkerboard Frame

Figure 15: The different kinds of generated Frames available

each other. This is useful if you want similar looking patterns. It is an optional parameter, but it is advised to always set it so frames that were generated once can be reused later, significantly reducing loading times.

- `side_length` - Only for the `'checkerboard'` type. An integer number that gives the side length of the squares that make up the checkerboard pattern. If the dimensions of the checkerboard frame are not divisible by the dimensions of the squares the frame will be cropped (the bottom and right will be cut off) to fit the dimensions.
- `wavelength` - The wavelength (in pixels) of the wave that modulates the grating pattern ie. the distance between the middle of the white or black lines.
- `angle` - An angle of the wave's travel in degrees. 0 will result in vertical lines that are rotated in the positive direction (clockwise) as you increase the number (eg. you can see 45° lines of Figures 15a and 15b).
- `filename` - Used only for `'image'` type Frames. The absolute path of the image file to be used. *Note: Make sure to use forward slashes in the filename instead of backslashes.*
- `zone_type` - An optional string that can be used to set an identifier for zone rules. If you leave it blank it defaults to `'generic'`.

Example:

```
# FRAMES
LVL.add_block('binarynoise', length=1000, random_seed=73, zone_type='noise')
LVL.add_block('wood', length=1000, random_seed=73, zone_type='neutral')
LVL.add_block('marble', length=1000, random_seed=73, zone_type='neutral')
LVL.add_block('greynoise', length=1000, random_seed=73, zone_type='noise')
LVL.add_block('square', length=1000, wavelength=200, angle=45, zone_type='right')
LVL.add_block('cloud', length=1000, random_seed=24, zone_type='neutral')
LVL.add_block('checkerboard', length=1000, side_length=45, zone_type='generic')
LVL.add_block('cloud', length=1000, random_seed=25, zone_type='neutral')
LVL.add_block('square', length=1000, wavelength=200, angle=0, zone_type='aversive')
LVL.add_block('cloud', length=1000, random_seed=26, zone_type='neutral')
LVL.add_block('square', length=1000, wavelength=200, angle=45, zone_type='right')
LVL.add_block('square', length=1000, wavelength=200, angle=315, zone_type='left')
LVL.add_block('cloud', length=1000, random_seed=27, zone_type='neutral')
```

Events

Events are things that can happen to the animal. The following table shows the different kinds you can define.

Name	Description	Parameters
`teleport`	Teleport to the given coordinate.	`target_position`
`random_teleport`	Teleports to the middle of a random zone with given type (excluding the current one).	`list_of_target_zones`
`port_on`	Pulls the given port high (eg. Opens a valve).	`port`
`port_off`	Pulls the given port low (eg. Closes a valve).	`port`
`start_burst`	Starts a burst pattern according to a given pattern on a given port.	`port, on_time, pause_time`
`stop_burst`	Stops the bursting on a given port.	`port`
`pause`	Pauses the simulation at a given position. While the simulation is paused, zone rules (see: next section) are inactive. Set position to None if you want to pause right where the animal is.	`pause_position`
`unpause`	Continues the simulation at a given position. Set position to None if you want to unpause right where the animal is.	`unpause_position`
`print`	Displays the given message in the console.	`message`

- ▷ `target_position, pause_position, unpause_position` - Integer numbers that gives a target position in pixels.
- ▷ `list_of_target_zones` - A list of strings with all the zone types that are valid targets for this teleport event (eg.: `['neutral', 'left', 'right']`).
- ▷ `port` - The digital output port used for this event. An integer number between 1 and 4.
- ▷ `on_time` - A parameter for bursting. The time for which the port will be pulled high in seconds.
- ▷ `pause_time` - A parameter for bursting. The time for which the port will be pulled low in seconds.
- ▷ `message` - The message that will be displayed. Must be a string.

Example:

```
# EVENTS
LVL.add_event('tp', 'random_teleport', ['right', 'left'])
LVL.add_event('start_puff', 'start_burst', 2, 0.1, 0.05)
LVL.add_event('stop_puff', 'stop_burst', 2)
LVL.add_event('pp', 'pause', 0)
LVL.add_event('up', 'unpause', None)
LVL.add_event('port_1_high', 'port_on', 1)
LVL.add_event('port_1_low', 'port_off', 1)
```

Rules

Rules are the third and last core component of Levels, these are what connect the Events to conditions so that you can define how you want the conditioning to happen. There are 3 different types you can use.

Rule type	Description	Parameters
`zone`	If the animal stays in the given zone for a given time the given event will be triggered.	`zone_type, delay`
`velocity`	If the velocity of the animal is above or below a certain threshold, the given event will trigger.	`vel_rule_type, threshold, delay`
`smooth_velocity`	Same as the velocity rule but a given number of velocities are averaged to smooth out the data.	`bin_size, vel_rule_type, threshold, delay`
`speed`	If the absolute sum of the last x velocities are above or below a certain threshold, the given event will trigger.	`speed_rule_type, threshold, bin_size`
`keypress`	Pressing the given key on the keyboard triggers the event.	`key`
`input`	Triggers when the given input rises, falls or changes.	`input_id, trigger_type`

- ▷ `zone_type` - The group of Frames you want this rule to trigger in.
- ▷ `delay` - How much time should pass (in seconds) before the rule is triggered, ie. how long should the animal stay in the zone or be above or below a velocity threshold.
- ▷ `threshold` - The threshold above or below which the rule triggers.
- ▷ `vel_rule_type, speed_rule_type` - Can be `'above'` or `'below'` depending on how you want to compare the animals movement to the threshold.
- ▷ `bin_size` - How many velocity points should be summed or averaged.
- ▷ `key` - The key on the keyboard that triggers the rule. Should be given as a string eg.: `'space'`, `'F8'`, `'NUM_5'`, `'A'`. Do not use `'ESCAPE'` as it is always set to exit the simulation.
- ▷ `input_id` - The number of the digital input on the device you want to use for the rule (1 or 2).
- ▷ `trigger_type` - If set to `'rise'` the rule will trigger when the state of the given input changes from low to high. If set to `'fall'` it triggers when the change is from high to low. If set to `'change'` it triggers in both cases.

To add a rule you can call your Level's `add_rule` command like so: `LVL.add_rule(rule_type, event, rule_parameters)`

Example:

```
# RULES
LVL.add_rule('zone', 'tp', 'neutral', 5)
LVL.add_rule('zone', 'start_puff', 'aversive', 3)
LVL.add_rule('zone', 'stop_puff', 'neutral', 0)
LVL.add_rule('speed', 'pp', 'above', 1000, 100)
LVL.add_rule('speed', 'up', 'below', 1000, 100)
LVL.add_rule('velocity', 'port_1_high', 'above', 20, 3)
LVL.add_rule('velocity', 'port_1_low', 'below', 15, 3)
```

The rules defined in this example will result in the following:

- If the mouse spends 5 or more seconds in one of the `'neutral'` zones it will be teleported out to a left or right zone.
- If the mouse spends 3 or more seconds in the aversive zone the valve connected to port B will start bursting as defined in the `start_puff` event.
- If the mouse enters a neutral zone the bursts of port B will stop immediately.
- The simulation will pause if the absolute sum of the last 100 velocities is above 1000 and unpause again if it gets below 1000.
- If the mouse runs with a higher velocity than 20 for at least 3 seconds, port A will be opened.
- If the mouse slows down and stays below 15 velocity, for at least 3 seconds port A will close.

Saving a preview image

When you finished making your level you can save a 1:1 ratio image of it with the `Level.save_image()` command (eg. `LVL.save_image()`). When the program is run you will be given a file selection window to give the location and filename of the image.

Saving a summary

You can save a human readable summary text with the `Level.save_summary()` command of your level. You will be presented with a file selection window to specify the location and filename of the summary file (eg. `LVL.save_summary()`).

Playing the Level

If you have a Level made you can call it's `play()` command to start the simulation. By default this would start the simulation with the primary monitor as the left screen, with the first gramophone as an input, with a gramophone speed to VR speed ratio of 1:1, with black and white colours on a single monitor and it would run until you press escape or close the simulation window. You can change the defaults by setting the corresponding, optional argument of the play command to something else.
The optional arguments are:

- ▷ `vel_ratio` - float number. The velocity read from the gramophone gets multiplied by this. Setting it to a negative value changes the forward direction. 1 by default.
- ▷ `runtime_limit` - float number. How long should the simulation run (in minutes).
- ▷ `left_monitor` - integer number or `None`. Which monitor is used as a left side display.
- ▷ `right_monitor` - integer number or `None`. Which monitor is used as a right side display.
- ▷ `gramophone_serial` - the serial number of the Gramophone device you wish to use. It can be given as a hexadecimal number eg.: `0xA0002`. If a device is not specified a random connected one will be selected automatically.
- ▷ `fullscreen` - `True` for full screen and `False` for windowed mode.
- ▷ `skip_save` - If set the `True` a save dialog for the log file will not be shown on start. Useful when testing levels.

Example:

```
LVL.play(left_monitor=None, right_monitor=1, vel_ratio=1280, runtime_limit=30,
    fullscreen=True)
```

Manual output control

During simulation you can manually control the outputs with Ctrl+numbers (eg. Ctrl+2 toggles digital output 2). Alternatively you can open the Recorder, connect to the same Gramophone and control the outputs there.
Note: The LinMaze window must be active for the keyboard shortcuts to work. Click in the window that is displaying patterns or select it on the taskbar to activate it.

Examples

A collection of examples are available on GitHub in the examples folder of the GramophoneTools repository. Alternatively you can open this folder of examples after installing the GramophoneTools package with the `gram examples` command.

Simulation results

When you call the play command you will be presented with a file selection window, on which you can select the location and filename of your log file. The log is automatically saved every 10 seconds or on the end of the simulation (on proper exit, do not close the console window, close the simulation window or select it and exit with the `Esc` key).

The log file is an HDF5 file with a `.vrl` extension. For more information about it's structure please visit the **online documentation of the LinMaze module**.