goxpyriment — Migration Guide

Claude Sonnet and Christophe Pallier

2026-06-04

Contents

Migrating to goxpyriment

This guide is for researchers who already know one of the mainstream experiment libraries and want to get productive in goxpyriment quickly. Each section maps the concepts and idioms of one tool to their goxpyriment equivalents, highlights the most important differences, and provides side-by-side code for a representative trial sequence.

- From Expyriment (Python)
- From PsychoPy (Python)
- From Psychtoolbox (MATLAB)

Before reading this guide, complete the Getting Started tutorial. It introduces the coordinate system, the rendering model, and the trial loop idiom that all three sections below assume.

From Expyriment (Python)

Goxpyriment was directly inspired by Expyriment and shares its philosophy: a single experiment object owns all resources, stimuli are objects with a `present`/`Show` method, and data are written to a structured file automatically. The mapping is the closest of the three tools covered here.

Concept map

Expyriment (Python)	goxpyriment (Go)	Notes
`design.Experiment("name")`	`control.NewExperimentFromFlags("name", bg, fg, fontSize)`	Also parses `-w`, `-d`, `-s` flags.
`control.initialize(exp)`	*(called inside `NewExperimentFromFlags`)*	SDL, audio, font, and data file initialized automatically.
`control.end(exp)`	`defer exp.End()`	Always defer immediately after construction.
`exp.clock.wait(1000)`	`exp.Wait(1000)`	Both pump the event loop; both abort on ESC.
`exp.clock.reset()`	`c := clock.NewClock()` then `c.Reset()`	`clock.NewClock()` starts a new reference; `c.Reset()` restarts it.
`exp.clock.time`	`c.NowMillis()`	Returns `int64` milliseconds.
`stim.present()`	`exp.Show(stim)`	Both do clear → draw → flip in one call.
`stimuli.TextLine("Hello", (0,0), exp.screen)`	`stimuli.NewTextLine("Hello", 0, 0, control.White)`	Position is center-relative in both (Expyriment center = (0,0), Y+ = up).
`stimuli.Circle(radius=50)`	`stimuli.NewCircle(50, color)`	
`stimuli.FixCross(size=30)`	`stimuli.NewFixCross(30, lineWidth, color)`	
`stimuli.Picture("img.png")`	`stimuli.NewPicture("img.png", x, y)`	
`exp.keyboard.wait()`	`exp.Keyboard.Wait()`	Returns `(Keycode, error)`.
`exp.keyboard.wait_for_keys([K_F, K_J])`	`exp.Keyboard.WaitKeys([]Keycode{K_F, K_J}, timeout)`	`timeout = -1` means no timeout.
`exp.keyboard.wait_for_keys(..., RT`	`exp.Keyboard.WaitKeysRT(keys, timeout)`	Returns `(key, rtMs, error)`.
`exp.data_variable_names = [...]`	`exp.AddDataVariableNames([]string{...})`	`subject_id` is prepended automatically — do not include it.
`exp.data.add([v1, v2, ...])`	`exp.Data.Add(v1, v2, ...)`	
`design.Block()`	`design.NewBlock("name")`	
`design.Trial()`	`design.NewTrial()`	
`trial.set_factor("cond", "target")`	`trial.SetFactor("cond", "target")`	
`trial.get_factor("cond")`	`trial.GetFactor("cond")`	
`block.add_trials([t1, t2])`	`block.AddTrial(t, copies, randomPosition)`	
`block.shuffle_trials()`	`block.ShuffleTrials()`	

Expyriment (Python)	goxpyriment (Go)	Notes
`io.DataFile`	`exp.Data` (`*io.DataFile`)	Opened automatically; output to `~/goxpy_data/`. `.csv` is the same CSV-with-comments format.

Side-by-side: simple RT trial

Expyriment

```
import expyriment
from expyriment import design, control, stimuli, io

exp = design.Experiment("SimpleRT")
control.initialize(exp)

fix   = stimuli.FixCross()
target = stimuli.Circle(radius=30, colour=(255,255,255))

exp.data_variable_names = ["rt"]

fix.present()
exp.clock.wait(500)
target.present()
key, rt = exp.keyboard.wait()
exp.data.add([rt])

control.end(exp)
```

goxpyriment

```
package main

import (
    "log"
    "github.com/chrplr/goxpyriment/control"
    "github.com/chrplr/goxpyriment/stimuli"
)

func main() {
    exp := control.NewExperimentFromFlags("SimpleRT", control.Black, control.White, 32)
    defer exp.End()

    fix    := stimuli.NewFixCross(30, 3, control.White)
    target := stimuli.NewCircle(30, control.White)

    exp.AddDataVariableNames([]string{"rt"})
```

```
    err := exp.Run(func() error {
        exp.Show(fix)
        exp.Wait(500)
        exp.Show(target)
        _, rt, err := exp.Keyboard.WaitKeysRT(nil, -1)
        if err != nil {
            return err
        }
        exp.Data.Add(rt)
        return control.EndLoop
    })
    if err != nil && !control.IsEndLoop(err) {
        log.Fatal(err)
    }
}
```

Key differences

The run loop. In Expyriment the trial logic runs in `main()` directly. In goxpyriment all SDL calls must run on the thread that owns the window; `exp.Run(func() error {...})` enforces this. Return `control.EndLoop` to exit cleanly.

Clock domains. Expyriment's `exp.clock.time` measures from `control.initialize`. `clock.NewClock()` measures from whenever you create it. For stimulus-onset-locked RT, use `exp.ShowTS` + `exp.Keyboard.GetKeyEventTS` (SDL nanosecond timestamps) rather than the `clock` package — see UserManual §6.

`.csv` files. The format is compatible: plain CSV with #-prefixed metadata lines. Existing Python scripts that read Expyriment data files with `pandas.read_csv(..., comment='#')` will read goxpyriment output without modification.

Adaptive staircases. Expyriment has no built-in staircase. Goxpyriment provides `staircase.NewUpDown` (Levitt 1971) and `staircase.NewQuest` (Watson & Pelli 1983), including a `Runner` for interleaved designs. Import `github.com/chrplr/goxpyriment/staircase`.

RSVP streams. Expyriment's `stimuli.extras.StreamingTextDisplay` is replaced by `stimuli.PresentStreamOfTexts` / `stimuli.PresentStreamOfImages`, which are VSYNC-locked, disable GC, and return a full `TimingLog` per item with nanosecond onset/offset timestamps.

From PsychoPy (Python)

PsychoPy Coder mode and goxpyriment are structurally similar: both give you a window object, stimulus objects, and explicit flip calls. The main adjustments are coordinate convention, the rendering model, and how timing is measured.

Concept map

PsychoPy (Python)	goxpyriment (Go)	Notes
`visual.Window(size=[1024,768], units='pix')`	`control.NewExperimentFromFlags()` with `-w` flag	Pass `-w 1024 768` for a 1024×768 window; default is fullscreen.
`win.flip()`	`exp.Flip()`	Both are VSYNC-locked. Usually use `exp.Show(stim)` instead.
`win.color = (-1,-1,-1)`	`control.Black` (background passed to constructor)	
`core.wait(0.5)`	`exp.Wait(500)`	PsychoPy uses seconds; goxpyriment uses milliseconds.
`core.Clock()` / `clock.getTime()`	`clock.NewClock()` / `c.NowMillis()`	See clock-domain note below.
`core.CountdownTimer(t)`	`c.SleepUntil(target)`	
`visual.TextStim(win, text="Hello")`	`stimuli.NewTextLine("Hello", x, y, color)`	
`visual.TextStim(..., wrapWidth=800)`	`stimuli.NewTextBox(text, width, pos, color)`	Word-wrapped.
`visual.Circle(win, radius=0.5, units='pix')`	`stimuli.NewCircle(radius, color)`	
`visual.Rect(win, width=100, height=50)`	`stimuli.NewRectangle(cx, cy, w, h, color)`	
`visual.ImageStim(win, image='img.png')`	`stimuli.NewPicture("img.png", x, y)`	
`visual.GratingStim(win, sf=0.05, ori=45)`	`stimuli.NewGaborPatch(sigma, theta, lambda, ...)`	See spatial/temporal frequency note below.
`stim.draw(); win.flip()`	`exp.Show(stim)`	
`stim.setPos((x, y))`	`stim.SetPosition(control.Point(x, y))`	
`event.waitKeys(keyList=['f','j'])`	`exp.Keyboard.WaitKeys(keys, timeout)`	
`event.waitKeys(maxWait=3)`	`exp.Keyboard.WaitKeys(keys, 3000)`	Timeout in ms.
`clock.getTime()` for RT	`exp.Keyboard.WaitKeysRT(keys, timeout)`	Returns RT in ms from call site.
`win.callOnFlip(clock.reset)` + `event.waitKeys`	`exp.ShowTS(stim)` + `GetKeyEventTS(...)`	Hardware-precise; no callOnFlip needed.
`data.TrialHandler(trialList, nReps)`	`design.NewBlock(...)` + `block.AddTrial(t, copies, true)`	
`data.ExperimentHandler(...)`	`exp.Data`	
`thisExp.addData("rt", rt)`	`exp.Data.Add(rt)`	

PsychoPy (Python)	goxpyriment (Go)	Notes
`data.QuestHandler(startVal, ...)`	`staircase.NewQuest(cfg)`	
`data.StairHandler(startVal, ...)`	`staircase.NewUpDown(cfg)`	
`sound.Sound('A', secs=0.5)`	`stimuli.NewTone(frequency, duration, volume)`	
`sound.Sound('beep.wav')`	`stimuli.NewSound("beep.wav")`	

Side-by-side: simple RT trial

PsychoPy

```
from psychopy import visual, core, event

win   = visual.Window([1024,768], color='black', units='pix')
fix   = visual.TextStim(win, text='+', height=30, color='white')
target = visual.Circle(win, radius=30, fillColor='white')

trial_clock = core.Clock()

fix.draw(); win.flip()
core.wait(0.5)

trial_clock.reset()
target.draw(); win.flip()
keys = event.waitKeys(keyList=['f','j'], timeStamped=trial_clock)
key, rt = keys[0]  # rt is in seconds

win.close()
```

goxpyriment

```
package main

import (
    "log"
    "github.com/chrplr/goxpyriment/control"
    "github.com/chrplr/goxpyriment/stimuli"
)

func main() {
    exp := control.NewExperimentFromFlags("SimpleRT", control.Black, control.White, 32)
    defer exp.End()

    fix    := stimuli.NewFixCross(30, 3, control.White)
    target := stimuli.NewCircle(30, control.White)
```

```
    err := exp.Run(func() error {
        exp.Show(fix)
        exp.Wait(500)
        onset, _ := exp.ShowTS(target)          // nanosecond timestamp at VSYNC flip
        key, eventTS, _ := exp.Keyboard.GetKeyEventTS(
            []control.Keycode{control.K_F, control.K_J}, -1,
        )
        rtMs := int64(eventTS-onset) / 1_000_000 // nanoseconds → milliseconds
        _ = key
        _ = rtMs
        return control.EndLoop
    })
    if err != nil && !control.IsEndLoop(err) {
        log.Fatal(err)
    }
}
```

Key differences

Units and coordinate system. PsychoPy supports multiple unit systems (norm, pix, deg, cm). Goxpyriment always uses pixels, with (0, 0) at the screen center, +X right, +Y **up** (same as PsychoPy's `units='pix'`). For visual-angle calculations, use the `units` package: `units.NewMonitor(widthCm, distanceCm, widthPx)` then `mon.DegToPx(deg)`.

Time units. PsychoPy uses seconds throughout (`core.wait(0.5)`, `clock.getTime()` returns float seconds). Goxpyriment uses **milliseconds** for `exp.Wait` and `WaitKeysRT`, and **nanoseconds** for hardware event timestamps (`ShowTS`, `GetKeyEventTS`). Divide nanosecond differences by `1_000_000` to get milliseconds.

`callOnFlip` is not needed. PsychoPy's `win.callOnFlip(clock.reset)` is a workaround for the fact that `flip()` and the clock reset are separate operations. `exp.ShowTS(stim)` captures the SDL nanosecond timestamp atomically at the VSYNC flip — no workaround needed.

Builder vs Coder. PsychoPy Builder generates Python Coder scripts. There is no Builder equivalent in goxpyriment; all experiments are written in code. The vibe-coding tip in the README describes how AI coding agents can generate Go experiment code from a plain-language description.

Spatial frequency. PsychoPy's `GratingStim` spatial frequency is in cycles/degree (when `units='deg'`). Goxpyriment's `spatialFreq` is in **cycles per pixel**. Convert: `sf_cpx = sf_cpd / mon.DegToPx(1)`.

TrialHandler nReps. PsychoPy's `TrialHandler` accepts a list of condition dicts and repeats them. The equivalent in goxpyriment is `block.AddTrial(t, nReps, randomPosition)` where `randomPosition=true` shuffles. For multiple condition lists, build one block per block and call `exp.ShuffleBlocks()` if needed.

Psychtoolbox-style timestamps. PsychoPy's `logging.flush()` and `win.recordFrameIntervals` give post-hoc timing diagnostics. In goxpyriment, `PresentStreamOfImages` returns a

`[]TimingLog` with per-item `ActualOnset`, `ActualOffset`, `OnsetNS`, `OffsetNS` — suitable for the same purpose.

From Psychtoolbox (MATLAB)

Psychtoolbox-3 (PTB) and goxpyriment share a lower-level philosophy: you open a window, draw into it, flip explicitly, and get a VBL timestamp back. The key differences are language ergonomics, the data-file model, and the adaptive staircase API.

Concept map

Psychtoolbox (MATLAB)	goxpyriment (Go)	Notes
`Screen('OpenWindow', screenNum, bgColor)`	`control.NewExperimentFromFlags(...)`	Also initializes audio, font, and data file.
`Screen('CloseAll')`	`defer exp.End()`	
`Screen('Flip', win)`	`exp.Flip()`	Both return a timestamp.
`Screen('Flip', win)` → VBL timestamp	`exp.ShowTS(stim)` → `uint64` nanoseconds	PTB returns seconds (64-bit float); goxpyriment returns nanoseconds (uint64).
`Screen('FillRect', win, color, rect)`	`stimuli.NewRectangle(cx, cy, w, h, color)` then `exp.Show(rect)`	
`Screen('DrawText', win, text, x, y, color)`	`stimuli.NewTextLine(text, x, y, color)` then `exp.Show(tl)`	
`Screen('MakeTexture', win, img)`	`stimuli.NewPicture("path", x, y)`	Texture is lazily uploaded on first `Draw`.
`Screen('DrawTexture', win, tex)`	`exp.Show(pic)`	
`Screen('DrawLine', ...)`	`stimuli.NewLine(x1, y1, x2, y2, color)` then `exp.Show(line)`	
`DrawFormattedText(win, text, 'center', 'center', color)`	`exp.ShowInstructions(text)`	Centered, waits for spacebar.
`WaitSecs(secs)`	`exp.Wait(ms)`	PTB uses seconds; goxpyriment uses **milliseconds**.
`GetSecs()`	`clock.GetTime()` (ms) or `clock.GetTimeNS()` (ns)	See clock-domain note below.
`KbWait`	`exp.Keyboard.Wait()`	
`KbCheck`	`exp.Keyboard.Check()`	Non-blocking poll.
`KbStrokeWait` / `KbName`	`exp.Keyboard.WaitKeys(keys, timeout)`	Pass `nil` for any key.

Psychtoolbox (MATLAB)	goxpyriment (Go)	Notes
RT via `GetSecs` before/after `KbWait`	`exp.Keyboard.WaitKeysRT(keys, timeout)` (ms)	Or `GetKeyEventTS` for nanosecond accuracy.
`PsychPortAudio('Open', ...)` / `'Start'`	`stimuli.NewSound(path)` + `snd.Play()`	
`MakeBeep(freq, dur, rate)`	`stimuli.NewTone(freq, duration, volume)`	
`Snd('Play', ...)`	`tone.Play()`	
`Quest('init', ...)`	`staircase.NewQuest(cfg)`	
`QuestUpdate(q, x, response)`	`sc.Update(correct)`	
`QuestQuantile(q)`	`sc.Intensity()`	
`QuestMean(q)`	`sc.Threshold()`	
Custom up-down staircase	`staircase.NewUpDown(cfg)`	Levitt (1971); 2-down-1-up built in.

Side-by-side: simple RT trial

Psychtoolbox

```
screenNum = max(Screen('Screens'));
[win, rect] = Screen('OpenWindow', screenNum, [0 0 0]);

cx = rect(3)/2;  cy = rect(4)/2;

% Fixation cross
Screen('DrawLine', win, [255 255 255], cx-15, cy, cx+15, cy, 3);
Screen('DrawLine', win, [255 255 255], cx, cy-15, cx, cy+15, 3);
Screen('Flip', win);
WaitSecs(0.5);

% Target circle
Screen('FrameOval', win, [255 255 255], [cx-30, cy-30, cx+30, cy+30], 3);
t0 = Screen('Flip', win);   % VBL timestamp in seconds

[~, t1, ~] = KbWait([], 2);
rt = (t1 - t0) * 1000;  % milliseconds

Screen('CloseAll');
```

goxpyriment

```
package main

import (
    "log"
    "github.com/chrplr/goxpyriment/control"
    "github.com/chrplr/goxpyriment/stimuli"
```

```
)

func main() {
    exp := control.NewExperimentFromFlags("SimpleRT", control.Black, control.White, 32)
    defer exp.End()

    fix    := stimuli.NewFixCross(30, 3, control.White)
    target := stimuli.NewCircle(30, control.White)

    err := exp.Run(func() error {
        exp.Show(fix)
        exp.Wait(500)
        onset, _    := exp.ShowTS(target)
        _, eventTS, _ := exp.Keyboard.GetKeyEventTS(nil, -1)
        rtMs := int64(eventTS-onset) / 1_000_000
        _ = rtMs
        return control.EndLoop
    })
    if err != nil && !control.IsEndLoop(err) {
        log.Fatal(err)
    }
}
```

Key differences

Time units. PTB uses **seconds** (double-precision float) throughout. Goxpyriment uses **milliseconds** for `exp.Wait` / `WaitKeysRT`, and **nanoseconds** for hardware event timestamps. To convert: multiply PTB seconds by 1000 to get ms, or 1,000,000,000 to get ns.

VBL timestamps. PTB's `Screen('Flip')` returns the VBL timestamp as `GetSecs` seconds. `exp.ShowTS(stim)` returns `sdl.TicksNS()` nanoseconds captured immediately after the VSYNC flip — a different clock origin. Do not mix SDL timestamps with `clock.GetTimeNS()` for RT calculation; use SDL timestamps exclusively (see UserManual §6).

Coordinate system. PTB uses a top-left origin (+Y down), consistent with screen pixel conventions. Goxpyriment uses a **center origin (+Y up)**, same as PsychoPy `pix` units. A stimulus at (0, 0) appears at screen center; a stimulus at (0, 200) appears 200 pixels **above** center.

Drawing model. PTB requires explicit `DrawLine`/`FillRect`/etc calls per frame and a separate `Flip`. Goxpyriment encapsulates each stimulus as an object; `exp.Show(stim)` does clear → draw → flip in one call. For multi-stimulus frames, draw each one (`stim.Draw(exp.Screen)`) then call `exp.Flip()` explicitly.

Data files. PTB has no built-in data file; researchers typically write custom CSV writers or use `fopen`/`fprintf`. Goxpyriment writes a structured `.csv` file automatically. Declare column names once with `exp.AddDataVariableNames`; append rows with

`exp.Data.Add(...)`.

Quest staircase. PTB's `Quest` functions are a set of global procedures (`QuestCreate`, `QuestUpdate`, `QuestMean`). `staircase.NewQuest(cfg)` is an object that implements the same Bayesian update; call `sc.Intensity()`, `sc.Update(correct)`, `sc.Threshold()`. The parameters (`tGuess`, `tGuessSd`, `pThreshold`, `beta`, `delta`, `gamma`) map directly.

EEG triggers. PTB sends triggers via `lptwrite` (Windows parallel port). Goxpyriment provides `triggers.NewDLPIO8`, `triggers.NewMEGTTLBox`, and `triggers.NewParallelPort`, all implementing the same `OutputTTLDevice` interface with `Send(mask)`, `Pulse(line, duration)`, and `AllLow()`. See UserManual §15.

No Flip scheduling. PTB's `Screen('Flip', win, when)` allows scheduling a flip at a specific VBL. Goxpyriment does not support scheduled flips; instead, use `clock.SleepUntil(target)` before `exp.Show` to achieve frame-accurate onset scheduling, or use the stream functions (`PresentStreamOfImages`) which handle VSYNC-locked scheduling internally.

Common gotchas (all three tools)

ESC handling. Goxpyriment treats ESC as a universal experiment abort. `exp.Wait`, `exp.Keyboard.WaitKeys`, and similar functions return `control.EndLoop` when ESC is pressed. Propagate this error upward from `exp.Run`'s callback; `control.IsEndLoop(err)` tests for it. There is no equivalent of PTB's `ListenChar`/`CharAvail` or PsychoPy's `event.clearEvents()` — use `exp.Keyboard.Clear()` to flush stale key presses before a new trial.

GPU texture allocation. The first time any visual stimulus is drawn, its SDL texture is allocated on the GPU. In all three tools the first presentation can be slower than subsequent ones. In goxpyriment, call `stimuli.PreloadVisualOnScreen(exp.Screen, stim)` (or `stimuli.PreloadAllVisual`) before the critical section to force allocation during an instruction screen, not during a timed trial.

RSVP / rapid stimulus sequences. Go's garbage collector can pause execution mid-sequence. The stream functions (`PresentStreamOfImages`, `PresentStreamOfTexts`) disable GC automatically for the duration of the stream. Do not implement your own VSYNC-locked loop without also disabling GC (`debug.SetGCPercent(-1)`).

Single binary distribution. Unlike Python (Expyriment, PsychoPy) and MATLAB (PTB), goxpyriment experiments compile to a single self-contained binary. Run `go build .` in the experiment directory; the result runs on any machine with the same OS and architecture without any runtime installation. For cross-platform distribution, use `GOOS=windows GOARCH=amd64 go build .` etc. — see Installation.