

Python Game Development III: Advanced

Engine-level game programming in Python: architecture and performance, game AI, procedural worlds and physics, GPU rendering and adaptive audio, networked multiplayer, and a studio capstone you playtest and ship.

FORMAT

Instructor-led

DURATION

14 sessions · 42 hrs

SESSION

3 h incl. a break

LEVEL

Advanced

HOW THE COURSE RUNS

Fourteen **three-hour sessions**, one module each. Every session has the same shape: a warm-up recap, two lessons (Session 14 is one studio lesson), a break, and a wrap-up with an exit ticket. Each lesson runs **frame → live-code → guided build → check**: the instructor frames the idea and live-codes the lesson's examples, then the room builds the Practice Exercise together up to a named step.

The lessons hold more than a room can do: each module is **3 to 3.5 hours of content** for a learner working alone, against about 2.5 hours of lesson time in the room. So the room does the framing, the demos and a **guided build**, and the remaining exercise steps, the Going Further challenges and the learning journal **finish at home**. Each session's schedule below names what the room covers; plan on **2 to 3 hours of homework** a week.

Every lesson lives on the free course site, with the complete example solution, a quick quiz and a journal prompt, so a missed session can be caught up at home. The course ends in a **studio capstone**: learners pitch their own game, build its vertical slice in Session 14, playtest it with classmates and ship it over three more milestones.

Weekly: 14 weeks, one session a week; the best fit for the homework load

Twice weekly: 7 weeks; ask for the at-home steps before the next session

Two terms: Sessions 1–7 (architecture, AI, worlds, physics, GPU), then 8–14 (atmosphere, networking, studio)

ENGINE Sessions 1–2 **GAME AI** 3–4 **WORLDS & PHYSICS** 5–6 **GRAPHICS & SOUND** 7–8
NETWORKING 9–11 **STUDIO** 12–14

SESSION SCHEDULE

Times assume a 9:00 start. Shift the whole block to fit your schedule (an afternoon or evening run works the same way).
Lesson titles link to the lesson pages on the course site.

SESSION 1 Engine Architecture

ENGINE

By end of session: structure a growing game as an entity-component-system (entities are ids, components are data, systems are behavior, run in a deliberate order) and decouple its systems with a typed event bus that survives a broken listener.

Deliverable: Component Arena (ECS) and Event Bus Playground

PART 1 Module 1 · Engine Architecture (I)

9:00–10:45

9:00 Welcome, environment check & lab setup · Course map (14 sessions), the pygame-ce check, copy the Module 1 starters, a predict-the-output dataclass recap.

9:15 **L01** Entity-Component-System · Live-code: the World and query(), systems in order, the Minimal ECS and the comet. Build: Component Arena Steps 1–5.

10:35 Break

PART 2 Module 1 · Engine Architecture (II)

10:45–12:00

10:45 **L02** Event Systems · Live-code: the minimal bus with a broken listener, the tie crash. Build: Event Bus Playground Steps 1–3.

11:50 Wrap-up & exit ticket · Order-bug relay, exit ticket, homework (at-home steps), bring list and a preview of Session 2.

SESSION 2 Performance & Robust Saves

ENGINE

By end of session: measure before optimizing with perf_counter, cProfile and timeit, fix the hot paths the numbers point at (including a spatial hash), and write save files with a checked header, CRC, version migrations and atomic writes.

Deliverable: Hot Path Hunt (with a before/after comment) and Bulletproof Save Slot

PART 1 Module 2 · Performance & Robust Saves (I)

9:00–10:42

9:00 Warm-up, recap & lab setup · Homework show-and-tell (the Hat Trick, a stunned enemy), a heap-order predict recap, copy the Module 2 starters, plug in.

9:12 **L03** Profiling & Performance · Phase timer, cProfile and timeit live runs, rendering fixes, the pair-count table. Build: Hot Path Hunt Steps 1–4.

10:32 Break

PART 2 Module 2 · Performance & Robust Saves (II)

10:42–12:00

10:42 **L04** Robust Saves: Integrity, Versioning, Security · Pickle demo, struct and the header, the container live, migrations, atomic writes. Build: Bulletproof Save Slot Steps 1–4.

11:50 Wrap-up & exit ticket · Header-reading relay, exit ticket, homework, bring list, preview Session 3.

SESSION 3

Movement AI

GAME AI

By end of session: steer agents with seek, flee, arrive and flocking as real accelerations, choose admissible heuristics and weighted A* from measurements, and move a crowd with one flow field.

Deliverable: Flock and Hawk and Crowd Commander

PART 1 Module 3 · Movement AI (I)

9:00–10:42

9:00 Warm-up, recap & lab setup · Homework show-and-tell (before/after numbers, a save attack), a Vector2 predict recap, copy the Module 3 starters.

9:12 **L05** Steering & Flocking · Units, seek/flee/arrive live, the three rules, blending, gauges. Build: Flock and Hawk Steps 1–5 plus a first toggle.

10:32 Break

PART 2 Module 3 · Movement AI (II)

10:42–12:00

10:42 **L06** Advanced Pathfinding · Admissibility, the overestimate demo, weighted A* live, JPS concept, flow fields. Build: Crowd Commander Steps 1–4.

11:50 Wrap-up & exit ticket · Units relay, exit ticket, homework, bring list, preview Session 4.

SESSION 4

Decision AI

GAME AI

By end of session: build reactive behavior trees on a blackboard that interrupt long actions on the next tick, and utility AI with response curves, personality and hysteresis that never flickers.

Deliverable: Worker Behavior Tree and Utility Critter

PART 1 Module 4 · Decision AI (I)

9:00–10:47

9:00 Warm-up, recap & lab setup · Show-and-tell (gauges, weight table), an Enum predict recap, copy the Module 4 starters.

9:12 **L07** Behavior Trees · Leaves, Sequence with memory, the reactive Selector, the toolkit live, decorators, Escape's hysteresis. Build: Worker Behavior...

10:37 Break

PART 2 Module 4 · Decision AI (II)

10:47–12:00

10:47 **L08** Utility AI · decide(), curves live, the negative-score trap live, personality once, hysteresis. Build: Utility Critter Steps 1–4 plus one...

11:52 Wrap-up & exit ticket · Tree-or-scores relay, exit ticket, homework, bring list, preview Session 5.

SESSION 5 Procedural Worlds

WORLDS & PHYSICS

By end of session: generate connected dungeons with binary space partitioning and caves with cellular automata, place a start and exit far apart, and tile road networks with a seeded wave function collapse.

Deliverable: Dungeon & Cave Viewer and WFC Road Builder

PART 1 Module 5 · Procedural Worlds (I)

9:00–10:37

9:00 Warm-up, recap & lab setup · Module 4 show-and-tell, a predict-the-output seed recap, copy both starters and run them.

9:15 **L09** Dungeons & Caves (BSP, Cellular Automata) · Live-code: scatter vs BSP, the cave rule, flood fill, two BFS sweeps. Build: Dungeon & Cave Viewer steps 1–4.

10:25 Break

PART 2 Module 5 · Procedural Worlds (II)

10:37–12:00

10:37 **L10** Wave Function Collapse · Live-code: the wave, entropy, observe/propagate, the terrain run, sockets. Build: WFC Road Builder steps 1–5.

11:50 Wrap-up & exit ticket · Seed relay, exit ticket, homework, preview Session 6 and its install.

SESSION 6 Physics Engines

WORLDS & PHYSICS

By end of session: collide rotating convex polygons with SAT and contact impulses with spin and friction, assemble a small fixed-step physics engine with joints, compare integrators, and rebuild the scene in pymunk.

Deliverable: Tumbling Crates, Mini Physics Engine and a pymunk comparison

PART 1 Module 6 · Physics Engines (I)

9:00–10:34

9:00 Warm-up, recap & lab setup · Seed match from Module 5, a Vector2 predict-the-output recap, pymunk import check, copy the starters.

9:12 **L11** SAT & Rotational Collisions · Live-code: SAT on three pairs, the normal flip, spin and inertia, impulses with a friction clamp, tunneling vs CCD. Build...

10:22 Break

PART 2 Module 6 · Physics Engines (II)

10:34–12:00

10:34 **L12** Build a Physics Engine (+ pymunk) · Live-code: engine anatomy, fixed step and exp damping, the integrator shoot-out, accumulated impulses and the joint, a pymunk...

11:50 Wrap-up & exit ticket · Physics bug clinic, exit ticket, homework, preview Session 7 and its GPU check.

SESSION 7 GPU Rendering

GRAPHICS & SOUND

By end of session: open an OpenGL 3.3 core context from pygame-ce, draw with moderngl buffers and shaders, upload Surfaces as textures, and chain framebuffer passes into a bloom pipeline.

Deliverable: Shader Sandbox and Bloom Pipeline

PART 1 Module 7 · GPU Rendering (I)

9:00–10:47

9:00 Warm-up, numpy recap & GL check · Module 6 callback, a numpy dtype predict, everyone runs the scaffold and reads their OpenGL version; triage anyone below 3.3.

9:15 **L13** moderngl Foundations · Live-code: the pipeline, the scaffold, GPU Hello, a full-screen quad with uniforms, textures, shader errors. Build: Shader...

10:35 Break

PART 2 Module 7 · GPU Rendering (II)

10:47–12:00

10:47 **L14** Post-processing · Live-code: framebuffers, ping-pong, a correct bloom, vignette, tone map and order. Build: Bloom Pipeline steps 1–4.

11:52 Wrap-up & exit ticket · Before/after bloom on the projector, exit ticket, homework, preview Session 8.

SESSION 8 Atmosphere

GRAPHICS & SOUND

By end of session: light 2D scenes with lightmaps and normal-mapped walls on the GPU, synthesize and filter sound with numpy, and fade layered music with the game's intensity.

Deliverable: Torchlit Wall and Adaptive Soundtrack

PART 1 Module 8 · Atmosphere (I)

9:00–10:45

9:00 Warm-up, recap & setup · Journal share, a Module 7 predict-the-output recap (the texture y-flip), copy the Module 8 starters, confirm moderngl opens a...

9:15 **L15** 2D Lighting · Live-code the lightmap (lightmap_solution.py), light shapes, normal maps, Lambert, GPU lights. Build: Torchlit Wall steps 1–4.

10:35 Break

PART 2 Module 8 · Atmosphere (II)

10:45–12:00

10:45 **L16** Adaptive Audio & DSP · Live-code the beep, synth recipes, echo / low-pass / clip (synth_dsp_solution.py keys 1–5), layers and fades. Build: Adaptive...

11:50 Wrap-up & exit ticket · Two-minute show-and-hear, exit ticket, homework, preview Session 9 (networking).

SESSION 9

Networking Foundations

NETWORKING

By end of session: reason about topologies, latency and TCP vs UDP, write echo servers on 127.0.0.1, frame messages with struct, and serve many clients with threads and locks.

Deliverable: Connection Scaling Visualizer and Framed Chat Server

PART 1 Module 9 · Networking Foundations (I)

9:00–10:45

9:00 **Warm-up, recap & setup** · Show-and-hear from Module 8, a "who owns the game?" warm-up, copy the Module 9 starters, open two terminals.

9:15 **L17** **Sockets, TCP & UDP** · Topologies, latency, TCP vs UDP (drop-packet demo), live-code the TCP and UDP echo servers. Build: First Echo, then Connection...

10:35 *Break*

PART 2 Module 9 · Networking Foundations (II)

10:45–12:00

10:45 **L18** **Framing & Concurrency** · The socketpair boundary demo, framing.py and its self-test, threads and locks, selectors. Build: Framed Chat Server steps 1–6...

11:50 **Wrap-up & exit ticket** · "Which lock guards what?" relay, exit ticket, homework, preview Session 10.

SESSION 10

Real-time Multiplayer

NETWORKING

By end of session: keep an authoritative server responsive with client prediction, reconciliation and interpolation, and judge hits fairly with lag compensation and a capped rewind.

Deliverable: Predict, Reconcile, Interpolate and Rewind and Hit

Also: Optional homework reading: State Sync & Bandwidth (deltas, quantization, interest management).

PART 1 Module 10 · Real-time Multiplayer (I)

9:00–10:47

9:00 **Warm-up, recap & setup** · One Module 9 result, a "trace the message" recap, copy the Module 10 starters (no sockets today).

9:12 **L19** **Client Prediction & Reconciliation** · Authoritative server, aliasing demo, LaggyLink, prediction, reconciliation, interpolation, the three-toggle demo. Build: Predict...

10:37 *Break*

PART 2 Module 10 · Real-time Multiplayer (II)

10:47–12:00

10:47 **L20** **Lag Compensation** · The delay table in ticks, history, state_at, the ray test, judge and favor-the-shooter, the fire demo. Build: Rewind and Hit...

11:52 **Wrap-up & exit ticket** · Exit ticket, homework (incl. the optional State Sync reading), preview Session 11.

SESSION 11

Lobbies & Matchmaking

NETWORKING

By end of session: build a lobby with rooms, ready checks and host migration and a pygame client that never freezes, then match players with Elo ratings, widening skill windows and ready checks.

Deliverable: Lobby Client and Matchmaking Simulator

PART 1 Module 11 · Lobbies & Matchmaking (I)

9:00–10:45

9:00		Warm-up, recap & setup · One optional-reading result, the lock relay from Module 9, copy the Module 11 starters and helpers.
9:15	L21	Lobby Server & Client · Protocol, the Room model, Room Rules warm-up (TODOs 1–4), the server demo, the reader-thread client. Build: Lobby Client steps...
10:35		Break

PART 2 Module 11 · Lobbies & Matchmaking (II)

10:45–12:00

10:45	L22	Matchmaking & Ratings · Fair or fast (window demo), Elo by hand, the matcher, quality and snake draft, ready checks. Build: Matchmaking Simulator steps...
11:50		Wrap-up & exit ticket · Networking-unit recap, exit ticket, homework, preview Session 12 (Genre Studio).

SESSION 12

Genre Studio

STUDIO

By end of session: give a top-down racer believable grip, understeer and oversteer, and build a compact real-time strategy game with per-faction economies, octile A*, production and research timed in seconds.

Deliverable: Lap Racer and Tiny RTS

PART 1 Module 12 · Genre Studio (I)

9:00–10:25

9:00		Warm-up, recap & lab setup · Elo recap from Module 11, the dot-product split that L23 depends on, copy both starters and run the racer.
9:12	L23	Racing Physics · Bicycle model live, brakes and drag, the Grip lab demo, tracks and laps. Build: Lap Racer to-dos 1–4, then the to-do 6 crash and...
10:15		Break

PART 2 Module 12 · Genre Studio (II)

10:25–12:00

10:25	L24	Real-time Strategy · Manhattan vs octile demo, factions and derived stats, a guided tour of one worker's trip. Build: Tiny RTS to-dos 1–7 (the Module...
11:50		Wrap-up & exit ticket · Seam hunt, exit ticket, homework, preview Session 13, bring list.

SESSION 13

Balance & Testing

STUDIO

By end of session: drive difficulty from one level with dynamic adjustment that respects the player, then plan and run a playtest with telemetry and report the results honestly.

Deliverable: Adaptive Arena and Instrumented Playtest; capstone design doc (at home)

PART 1 Module 13 · Balance & Testing (I)

9:00–10:30

- 9:00 Warm-up, recap & lab setup · Two Module 12 results, a predict-the-curve hook, copy both starters and run the arena.
- 9:12 **L25** Difficulty & DDA · Flow channel, the DDA simulator, one level many knobs, the assist bug, DDA as a control loop, flow mode and rubber banding...
- 10:20 Break

PART 2 Module 13 · Balance & Testing (II)

10:30–12:00

- 10:30 **L26** Playtesting & Telemetry · Protocol and consent, three telemetry channels, CSV done right, metrics and SUS, the permutation test. Build: Instrumented...
- 11:45 Wrap-up, capstone launch & exit ticket · Exit ticket, homework, hand out the design doc, explain Session 14's one-minute pitch, bring list.

SESSION 14

Capstone Studio

STUDIO

By end of session: pitch a scoped game with two verifiable Advanced systems, build its vertical slice on the studio skeleton, run a silent peer playtest, fix the top issue, and plan the release.

Deliverable: Capstone Milestone 1 (a playable vertical slice) and a dated milestone plan

PART 1 Module 14 · Capstone Studio (I)

9:00–10:08

- 9:00 **LAB** **Pitch round** · One minute each, design doc on screen. The room asks: "How will you know each system works?" Push back on scope.
- 9:20 **L27** Build toward Milestone 1 (part 1) · Five-minute skeleton tour, then to-dos 1–4 on studio_starter.py. Copy a working version every 20 minutes.
- 10:00 Break

PART 2 Module 14 · Capstone Studio (II)

10:08–12:00

- 10:08 **L27** Build toward Milestone 1 (part 2) · Real win/lose rule, events published (to-do 5). Gate at 10:15: a playable loop, title to game over.
- 10:30 **L27** Peer playtest rounds · 8 minutes per tester, silent observation, think-aloud, telemetry on; 2-minute debrief after each.
- 11:15 **L27** Triage and fix the top issue · Triage table; one change, justified by notes or telemetry; test the fix.
- 11:45 Show and tell, milestone plan & course close · Thirty-second demos, Milestones 2–4 dated, rubric explained, course close and exit ticket.

COURSE DETAILS

■ Requirements

- ◆ Completed **Python Game Development II** (or equal pygame-ce experience) and **Intermediate Python**: dataclasses, abstract base classes, type hints, file I/O. Threads and locks are taught in Sessions 9–11; numpy gets a primer in Session 7.
- ◆ A laptop (Windows, macOS or Linux) you can install software on, with **Python 3.10+**, **pygame-ce** (not pygame), **moderngl**, **numpy** and **pymunk**
- ◆ A graphics chip and driver with **OpenGL 3.3 core** for Sessions 7–8 (most laptops from the last ten years, including Apple silicon); a physical machine, not a VM or remote desktop
- ◆ Permission to run programs that open network sockets on **127.0.0.1** (Sessions 9–11); nothing is opened to the internet
- ◆ VS Code or another code editor, and about 2–3 hours a week for the at-home steps

■ What's Provided

- ◆ Free lesson pages for all 27 lessons plus one optional reading, each with examples, a practice exercise, an example solution and a quick quiz
- ◆ Starter files for every practice exercise, and a **setup check** script to run before Session 1
- ◆ A participant workbook: key ideas, worked examples, predict-then-run boxes, self-check quizzes and journal prompts
- ◆ The capstone kit: a one-page design-doc template, a studio skeleton with scenes, a fixed timestep, events, safe saves and telemetry, and a sample capstone
- ◆ Instructor demos and feedback on exit tickets, playtests and the capstone

■ What to Bring

- ◆ **Every session:** your laptop and charger, the workbook, and your finished files from the session before
- ◆ **S3:** a mouse if you have one (a trackpad works) · **S5, S10:** a pencil
- ◆ **S6:** pymunk installed · **S7:** moderngl and numpy installed and the OpenGL check passing, on a physical machine
- ◆ **S8:** headphones or earbuds · **S9–S11:** two terminal windows side by side
- ◆ **S14:** your filled-in capstone design doc, the libraries your game needs, the lab files you'll reuse, and any free (CC0) assets with their license notes

■ Assessment

- ◆ **Self-checks:** six questions per lesson in the workbook (eight for the capstone lesson), plus each lesson's quick quiz (practice, not graded)
- ◆ **Practice exercises:** two per session (Session 14 is the capstone build), each with a clear "done when" list and automated tests the instructor can run
- ◆ **Exit tickets:** one short question at the end of every session
- ◆ **Capstone:** a complete game with at least two Advanced systems, a peer playtest and a playtest-driven change, and a build someone else ran; scored on the lesson's rubric (5 rows × 4 points) by you, a peer and the instructor

What You'll Be Able to Do

By the end of the course you can build and reason about the systems inside real game engines, in Python:

- ✓ Structure a game as an ECS with an event bus, and profile it before optimizing
- ✓ Steer and path crowds, and make NPCs decide with behavior trees and utility AI
- ✓ Collide rotating bodies and build a small physics engine, then use pymunk
- ✓ Build networked games with framing, prediction, lag compensation, lobbies and matchmaking
- ✓ Write save files that survive damage, old versions and crashes
- ✓ Generate reproducible dungeons, caves and tile worlds from a seed
- ✓ Render with GPU shaders, post-processing and 2D lighting, and make audio adapt
- ✓ Balance with DDA, playtest with telemetry, and ship a game others can run

© 2026 Ray de la Paz. Python Game Development III: Advanced. All rights reserved.

Technical trainer, full-stack developer, and course developer with 30+ years of experience · [Course home](#)

v1.0 · 14 sessions · 42 hours