# Game Engine Roadmap (Aug 2025 – Aug 2026)

This roadmap provides a week-by-week breakdown for developing a modular game engine project. It is structured to span approximately 50 weeks, ending in August 2026. The project emphasizes multi-language flexibility, interdisciplinary learning, and portfolio-ready artifacts at each stage.

## Phase 1 (Weeks 1–8): Core Engine Foundation

### Stage 1 (Toy System) Weeks 1-5

* ~~Week 1: Set up project repo + dev environment (C++/Rust + build system). Pick graphics library (SFML/SDL/OpenGL wrapper).~~
* ~~Week 2: Render a window with a solid color background. Handle window close events.~~
* ~~Week 3: Add input system (keyboard + mouse events). Log input for debugging.~~
* ~~Week 4: Draw basic shapes (rectangles, circles) on screen. Move a shape with keyboard input.~~
* ~~Week 5: Implement Entity-Component-System (ECS) skeleton. Define entities, components, and systems.~~

### Stage 2: Core Systems (Week 6)

* Week 6: Create a 'render system' in ECS to draw entities. Create a 'movement system' to update positions.
  + Create a **System base** class or concept.
  + Implement **MovementSystem**:
  + Loops through all entities with Position + Velocity.
  + Updates positions based on velocity.
  + Implement **RenderSystem**:
  + Loops through all entities with Position + a visual component (like RectangleShapeComponent).
  + Draws them to the window.
  + Goal: Entities can now **move and be drawn** through the ECS, instead of manually in main().

### Stage 3: Sprites & Assets (Week 7)

* Week 7: Add simple sprite rendering (load and draw PNG). Organize project folders (core, graphics, input, ECS).
  + Add **SpriteComponent** (stores sf::Sprite or texture info).
  + Load PNG textures from files.
  + Update **RenderSystem** to draw sprites instead of basic shapes.
  + Organize project structure:
  + core/ → ECS + engine core
  + graphics/ → sprites, shapes, rendering helpers
  + input/ → InputManager
  + Goal: ECS handles **visual entities** with sprites, ready for more game logic.
* Week 8: Create a 'scene manager' (switch between different screens). Demo: basic 'menu screen' and 'game screen.'

## Phase 2 (Weeks 9–16): Math & Physics

* Week 9: Build vector and matrix math utilities.
* Week 10: Implement transformations (translate, rotate, scale).
* Week 11: Add basic collision detection (AABB).
* Week 12: Extend collisions to circles and polygons.
* Week 13: Implement simple physics (gravity + velocity).
* Week 14: Add collision response (bouncing, stopping).
* Week 15: Implement a physics system in ECS.
* Week 16: Demo: objects fall, collide, and bounce within a scene.

## Phase 3 (Weeks 17–24): AI & Algorithms

* Week 17: Implement graph data structure for navigation.
* Week 18: Implement pathfinding (BFS, DFS).
* Week 19: Add A\* pathfinding.
* Week 20: Integrate NPC movement along paths.
* Week 21: Implement behavior trees for decision making.
* Week 22: Add state machines for NPC actions.
* Week 23: Build a simple dialogue system (rule-based).
* Week 24: Demo: NPC that moves and reacts to player input.

## Phase 4 (Weeks 25–32): Database & Persistence

* Week 25: Research database integration (SQLite or JSON).
* Week 26: Build asset manager for textures/sounds.
* Week 27: Add save/load functionality (JSON serialization).
* Week 28: Extend save/load to game states.
* Week 29: Store player stats and progress.
* Week 30: Add a logging system (performance + debugging).
* Week 31: Optimize data access with indexing/caching.
* Week 32: Demo: Game with save/load and asset management.

## Phase 5 (Weeks 33–40): Networking & Systems

* Week 33: Learn sockets and client/server basics.
* Week 34: Implement a local multiplayer setup.
* Week 35: Add basic chat system between clients.
* Week 36: Sync player positions across network.
* Week 37: Handle lag with interpolation.
* Week 38: Add simple server authoritative model.
* Week 39: Store/retrieve multiplayer stats in database.
* Week 40: Demo: simple online multiplayer prototype.

## Phase 6 (Weeks 41–50): Tools & Scripting

* Week 41: Research embedding Lua/Python scripting.
* Week 42: Implement scripting interface for entities.
* Week 43: Allow level logic in scripts.
* Week 44: Design basic editor UI (Java/C#).
* Week 45: Add entity placement in editor.
* Week 46: Link editor with engine runtime.
* Week 47: Add visual scripting prototype.
* Week 48: Polish editor features (save/load levels).
* Week 49: Integrate all modules (physics, AI, DB, networking).
* Week 50: Final demo + documentation + portfolio showcase.