Pixels, Text, and Go_

A Hobbyist's Guide to Making Games

Alan Bunjevac
October 2025

What We'll Cover_

- Why Go for game development?
- Game architecture patterns
- 2D graphics with Go
- FPS and VSync
- Game loop basics and physics
- Entity Component System
- Console-based games

Why Go for Game Development?_

- **Why not?**
- Simple, readable syntax
- Cross-platform compilation
- Growing ecosystem of game libraries
- Fast compilation times
- Great concurrency model (**YAGNI**)

"Retained" vs "Immediate" Mode_

Architectural patterns for game development.

Retained-mode GUI_

- Toolkit remembers the widget tree (buttons, labels, layouts)
- On change, it computes regions to repaint and issues draw calls
- Full redraw rarely required; updates stay localized
- Standard desktop applications

Immediate-mode GUI_

- Application calls UI code every frame inside the main loop
- Library re-issues draw commands each pass, mirroring a game loop
- Still event-driven, but optimized for rapid, per-frame updates
- Game-engine UIs / Dashboards / TUIs

Frames Per Second (FPS)_

- FPS counts how many frames you render each second
- Higher FPS means smoother visuals and lower input latency
- Dropped FPS often comes from heavy simulations, large draw calls, or blocking I/O on the main thread
- Most consoles and PC titles target 60 FPS; cinematic or resource-heavy games may ship at 30 FPS

$$1F = 0.016667s \approx 16.7ms$$

Understanding VSync_

Vertical Synchronization - synchronizing frame rendering with display refresh.

VSync on CRT Monitors_

The electron beam physically scans from top to bottom, then returns to the top (vertical blank/retrace).

If you swap the framebuffer mid-scan, you get "tearing" - the top half shows the old frame, bottom half shows the new frame.

VSync waits for the vertical blank interval before swapping buffers, ensuring a complete, tear-free image.

VSync on Modern Monitors_

LCD/LED panels refresh at fixed intervals (60Hz, 144Hz, etc.). The display controller reads from video memory at regular intervals.

Without VSync, the GPU may update the buffer while the display is reading it, causing tearing. VSync still synchronizes buffer swaps with the display's refresh cycle.

Modern alternatives: G-Sync, FreeSync (variable refresh rate).

Game Loop Basics_

```
const targetFPS = 60
const frameTime = 1.0 / targetFPS

state := initialState()

for {
    frameStart := now()

    input := pollInput()
    state = update(state, input, frameTime)

    render(state)

    elapsed := now() - frameStart
    sleep(max(0, frameTime - elapsed))
}
```

Frame Step Physics_

- Each update integrates over a timestep `Δt`, scaling movement and animation
- Velocity integration: `position = position + velocity * Δt`
- Keep `Δt` clamped (e.g., `min(Δt, 1/60)`) to avoid tunneling after long pauses

$$p(t + \Delta t) = p(t) + v(t) \cdot \Delta t$$

Acceleration Integration_

- Apply acceleration to velocity before updating position: `velocity = velocity + acceleration * Δt`
- Use forces or input to compute `acceleration` each frame, accumulate gravity and drag
- Cap maximum velocity to keep physics stable when `Δt` spikes

$$v(t + \Delta t) = v(t) + a(t) \cdot \Delta t$$

2D Graphics with Go_

- **ebiten** - Simple 2D game library
- **pixel** - Hand-crafted 2D game library
- **raylib-go** - Go bindings for raylib
- **go-sdl2** - Go bindings for SDL2 👈
- **Awesome Go** https://awesome-go.com

SDL2 - Simple DirectMedia Layer_

- **https://libsdl.org** - Official SDL website
- **github.com/veandco/go-sdl2** - Go bindings for SDL2

```
sudo apt install libsdl2-dev libsdl2-ttf-dev libsdl2-image-dev libsdl2-mixer-dev
```

- **SDL3**

Entity Component System (ECS)_

- Opposite to inheritance-style hierarchies
- **Entities** are lightweight IDs, **components** hold data, and **systems** operate on entities that match component filters
- Separating data from behavior improves cache locality and makes features like networking or replay recording easier
- Excels when you have many similar actors (bullets, particles, enemies) that share logic but differ by data

Console-based Games_

- **gdamore/tcell** - Rich terminal handling and input 👈
- **charmbracelet/bubbletea** - Elm-inspired stateful TUIs
- **charmbracelet/lipgloss** - Layout and styling for TUIs
- 🦀 Roguelike Tutorial (https://bfnightly.bracketproductions.com)

Resources_

- **Games with Go | Jack Mott** https://www.youtube.com/watch?v=9D4yH7e_ea8 🙏
- **Endless Sky** https://endless-sky.github.io/
- **Open Game Art** https://opengameart.org/
- **Dreamhold** https://eblong.com/zarf/zweb/dreamhold/
- **Awesome Go** https://awesome-go.com

Thank You!_

Questions...