Demoscene, Terminal, CMD, PowerShell, and DOSBox Code

A brutally practical list of repositories and source code you can learn from by editing and running them.

Author: Pablo Murad

April 2026

65 curated links - games, ASCII animations, shaders, demo tools, DOS retrocoding, and terminal engines.

How to Use This Document

- Start with items 1 to 6. They give you vocabulary and help prevent you from becoming a repository collector who never produces anything.

- Then choose one track: PowerShell/CMD for fast visible results, DOSBox for low-level work, or shaders for modern demoscene practice.

- Clone one repository at a time, run it as-is, change one visible variable, and then rewrite one small effect from scratch.

- Do not run random binaries outside a sandbox. For old demos, prefer DOSBox, a VM, or code you compiled yourself.

Honest Shortcut

If you try to learn everything at once, you will learn nothing. Use this order: 1) PowerShell Matrix, 2) clippy/ascii-animator, 3) ASCII Donut, 4) Bonzomatic + The Book of Shaders, 5) dos-dev-template + Tetris Assembly, 6) Second Reality only after that.

Recommended Learning Roadmap

Phase 1 - Terminal Basics

CursorPosition, ANSI colors, screen clearing, timing loops, and non-blocking input. Projects: Matrix.ps1, clippy, ascii-animator.

Phase 2 - Visual Mathematics

Sine, cosine, 2D/3D rotation, projection, and simple z-buffers. Projects: ASCII Donut, AsciiCube, ascii-3d-cube.

Phase 3 - Console Games

Game loops, collision, state, score, keyboard input, and text sprites. Projects: dotnet-console-games, CSharpConsoleGames, tetrigo.

Phase 4 - DOSBox and Low-Level Work

Mode 13h, video memory, BIOS interrupts, 8086 Assembly, C, and DJGPP. Projects: masm-tasm, dos-dev-template, dos3d, Tetris Assembly.

Phase 5 - Shader and Modern Demoscene Work

GLSL, fragment shaders, SDFs, raymarching, and music synchronization. Projects: The Book of Shaders, Bonzomatic, SDF, Rocket.

Phase 6 - Production

Build a short 30-to-60-second demo: intro, three effects, music or beep/track, credits, packaging, and a recorded video.

Base Commands

PowerShell:

```
git clone REPOSITORY_URL
cd folder
Get-ChildItem
.\script.ps1
```

CMD / Node.js / Python:

```
git clone REPOSITORY_URL
cd folder
python main.py
node index.js
```

DOSBox:

```
mount C C:\dosprojects
C:
cd PROJECT
program.exe
```

Curated Code and Repository List

Each item below includes a suggested execution path and what you should try to learn. Blue links are clickable.

0 - Start Here

01. Teach Yourself Demoscene in 14 Days

Runs on: Reading + guided projects Language: Markdown / references Level: Beginner

Description: A direct guide for entering the demoscene without getting lost in tooling. Use it as your main track before diving into harder source code.

Learn: demoscene vocabulary, demo categories, coder/artist/musician roles, and the mental pipeline

Open code/repository
`https://github.com/psenough/teach_yourself_demoscene_in_14_days`

02. Demoscene Starter Kits

Runs on: Windows, Linux, Processing, varied platforms Language: Processing / examples Level: Beginner

Description: A repository built to give you practical starting points. It is ideal for moving from simply watching demos to actually changing something.

Learn: boilerplate, visual loops, first effects, and the minimum structure of a demo

Open code/repository
`https://github.com/anttihirvonen/demoscene-starter-kits`

03. Awesome Demoscene

Runs on: Study map Language: Curated list Level: All levels

Description: A large index of demoscene resources. It is not something to run; it is something to mine. Treat it as the master list.

Learn: where to find engines, productions, tools, fonts, and communities

Open code/repository
`https://github.com/psykon/awesome-demoscene`

04. in4k.github.io

Runs on: Browser / documentation Language: Markdown / HTML Level: Intermediate

Description: Classic material for understanding tiny intros. The mindset is simple: every byte must justify its existence.

Learn: sizecoding, 1k/4k intros, and tricks for reducing file size

Open code/repository
`https://github.com/in4k/in4k.github.io`

05. 64k Scene

Runs on: Browser / documentation Language: HTML / JavaScript Level: Intermediate

Description: A gallery and documentation hub for 64k intros. Use it to understand the quality bar and decide what to study next.

Learn: 64k intros, procedural generation, audio synthesis, and audiovisual composition

Open code/repository
`https://github.com/64k-scene/64k-scene.github.io`

06. The Book of Shaders

Runs on: Browser / GLSL editor Language: GLSL / JavaScript Level: Beginner to advanced

Description: The cleanest bridge between visual mathematics and modern demoscene work. Run it in the browser, edit values, break things, and recover.

Learn: fragment shaders, noise, shapes, patterns, and introductory raymarching

Open code/repository
https://github.com/patriciogonzalezvivo/thebookofshaders

1 - Real Demoscene and Intros with Source

07. Second Reality - Future Crew

Runs on: DOSBox for the binary; historical source code for study Language: Assembly / C / Pascal Level: Advanced

Description: The source code for one of the most important MS-DOS demos ever made. Do not start here, but return to it when you want to study real root-level demoscene work.

Learn: VGA, 2D/3D effects, music synchronization, and modular organization in a large demo

Open code/repository
https://github.com/mtuomi/SecondReality

08. Oscar's Chair

Runs on: Windows / C++ build Language: C++ / GLSL Level: Advanced

Description: A winning 4k intro from a major event. Good for studying how a rich scene can come from little code and a lot of mathematics.

Learn: 4k intros, raymarching, and compressing an entire scene into a small mental model

Open code/repository
https://github.com/demoscene-source-archive/oscar-s-chair

09. Terrarium

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: Another strong small-intro study. The focus is learning how visuals are generated without traditional assets.

Learn: procedural nature, composition, and code-generated texture

Open code/repository
https://github.com/demoscene-source-archive/terrarium

10. Ohanami

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: Useful for comparing artistic decisions under brutal size constraints.

Learn: procedural aesthetics, compact animation, and minimalist shader work

Open code/repository
https://github.com/demoscene-source-archive/ohanami

11. Love Reaction

Runs on: Windows / C++ build Language: C++ / GLSL Level: Advanced

Description: A small intro with source code. Use it to study the structure of a complete production, not just an isolated effect.

Learn: transitions, generated scenes, and resource compression

Open code/repository
https://github.com/demoscene-source-archive/love-reaction

12. Horizon Machine

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: Useful for understanding how to sell scale and atmosphere with very little code.

Learn: landscapes, distance, procedural lighting, and short-form visual storytelling

Open code/repository
`https://github.com/demoscene-source-archive/horizon-machine`

13. Glittermorphosis

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: A good reference for transformation and glow effects within the vocabulary of intros.

Learn: morphing, particles, and compact visual style

Open code/repository
`https://github.com/demoscene-source-archive/glittermorphosis`

14. Dropletia

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: If you want to study liquid effects without a huge engine, this is the kind of source code that matters.

Learn: water, materials, and procedural surfaces

Open code/repository
`https://github.com/demoscene-source-archive/dropletia`

15. Alive Here Now, Forever

Runs on: Windows / C++ build Language: C++ / GLSL Level: Advanced

Description: Intro source code for studying aesthetics and engineering together.

Learn: short audiovisual composition, synchronization, and proceduralism

Open code/repository
`https://github.com/demoscene-source-archive/alive-here-now-forever`

16. Appear - Jetlag

Runs on: Windows / C++ build Language: C++ / shader Level: Advanced

Description: A strong piece for studying a more modern 4k/intro style.

Learn: shader intros, build structure, and compressed scenes

Open code/repository
`https://github.com/w23/jetlag_appear`

17. Crawlspace

Runs on: Windows / build according to the repository Language: Assembly / C / shader Level: Advanced

Description: Material for when you want to suffer productively. A 1k intro is sharpened-knife training.

Learn: 1k intros, byte obsession, and minimum viable tricks

Open code/repository
`https://github.com/in4k/crawlspace`

18. Star Traveler

Runs on: Browser / local server / optional Node.js Language: JavaScript Level: Intermediate

Description: A great bridge between demoscene thinking and JavaScript: easier to edit than old C++ toolchains.

Learn: small JavaScript demos, canvas/WebGL, and compact animation

Open code/repository
`https://github.com/depp/demo-traveler`

2 - Demo, Shader, and Audio Tools

19. Farbrausch Public Repo

Runs on: Windows / old Visual Studio or adaptation Language: C++ Level: Advanced

Description: A raw treasure chest of Farbrausch tools. Do not expect a friendly tutorial. This is heavy, valuable archaeology.

Learn: historical tooling, kkrunchy, Werkkzeug, the V2 synth, and demo pipelines

Open code/repository
`https://github.com/farbrausch/fr_public`

20. Bonzomatic

Runs on: Windows, Linux, macOS; builds with CMake Language: C++ / GLSL / HLSL Level: Intermediate

Description: One of the best ways to practice shaders like a demoscener: write, recompile, and see the result immediately.

Learn: shader live coding, real-time fragment shaders, and fast visual feedback

Open code/repository
`https://github.com/Gargaj/Bonzomatic`

21. Rocket

Runs on: Windows / Linux; editor + C library Language: C / C++ / Qt Level: Intermediate

Description: A classic tool for synchronizing visual parameters with music. Essential for understanding demos as audiovisual pieces.

Learn: music-to-visual synchronization, timelines, and keyframes

Open code/repository
`https://github.com/rocket/rocket`

22. Shader Minifier

Runs on: Windows; terminal/CMD; can be integrated into builds Language: F# / shader tooling Level: Advanced

Description: A tool for reducing shader code without changing behavior. It teaches how demosceners squeeze visual code.

Learn: GLSL/HLSL minification, preparation for 4k/64k, and build pipelines

Open code/repository
`https://github.com/laurentlb/Shader_Minifier`

23. SDF - Signed Distance Field Resources

Runs on: Reading + Shadertoy examples Language: GLSL / references Level: Intermediate

Description: A useful collection for learning the technique behind many modern demos: modeling shapes with mathematical functions.

Learn: raymarching, distance fields, and procedural geometry

Open code/repository
`https://github.com/CedricGuillemet/SDF`

24. Dwitter

Runs on: Browser / Python Django for the server Language: JavaScript / Python Level: Intermediate

Description: The idea is to create tiny visual demos. Excellent training for visual synthesis and code economy.

Learn: micro-demos in JavaScript, visual code golf, and very short animations

Open code/repository
`https://github.com/lionleaf/dwitter`

25. frame.js

Runs on: Browser / optional Node.js Language: JavaScript Level: Intermediate

Description: A JavaScript editor/framework for organizing animations. Useful as a browser lab for small demos.

Learn: sequencing, visual timelines, and browser animation

Open code/repository
`https://github.com/mrdoob/frame.js`

26. Axiom

Runs on: Windows / macOS; C++ build Language: C++ Level: Advanced

Description: Audio is code too. This repository helps explain procedural sound synthesis, a central ingredient in tiny intros.

Learn: node-based synthesis, procedural audio, and sound for demos

Open code/repository
`https://github.com/monadgroup/axiom`

27. OpenMPT / libopenmpt

Runs on: Windows; openmpt123 in the terminal Language: C++ Level: Intermediate

Description: A tracker and module playback library. If you want retro/demoscene sound, you need to understand this ecosystem.

Learn: trackers, MOD/XM/S3M/IT, and module playback in demos and games

Open code/repository
`https://github.com/OpenMPT/openmpt`

3 - PowerShell, CMD, and Terminal Animation

28. PowerShell Matrix Animation

Runs on: PowerShell 5.1+ / Windows Terminal / CMD calling pwsh Language: PowerShell Level: Beginner

Description: Start here if you want something that runs directly in PowerShell. Edit density, speed, renderer behavior, and characters.

Learn: cursor control, colors, parameters, and terminal render loops

Open code/repository
`https://github.com/avdaredevil/matrix`

29. fleschutz PowerShell Scripts

Runs on: PowerShell on Windows, Linux, or macOS Language: PowerShell Level: Beginner

Description: A huge collection of scripts. For terminal aesthetics, look for image-to-ASCII conversion, sound, and interactive scripts.

Learn: standalone .ps1 scripts, ASCII, sound, utilities, and automation

Open code/repository
`https://github.com/fleschutz/PowerShell`

30. clippy

Runs on: CMD on Windows; standard Python Language: Python Level: Beginner

Description: Excellent for learning the basic principle: every text file is a frame. Edit the art without fighting complex graphics.

Learn: ASCII frames stored as text files, simple animation, and terminal screensavers

Open code/repository
`https://github.com/con-dog/clippy`

31. ascii-animator

Runs on: PowerShell/CMD with Python Language: Python Level: Beginner to intermediate

Description: A tool and library for turning GIFs into ASCII animations. It teaches the pipeline from image to text to time.

Learn: GIF-to-ASCII conversion, animations, Game of Life, bars, and equalizers

Open code/repository
`https://github.com/soda480/ascii-animator`

32. Terminal Media Player - tplay

Runs on: Windows/Linux/macOS terminal Language: Go Level: Intermediate

Description: An ASCII media player. Study how it maps pixel luminance to characters and draws quickly in the terminal.

Learn: image, video, and webcam output in ASCII, plus fast terminal rendering

Open code/repository
`https://github.com/maxcurzi/tplay`

33. CMatrix

Runs on: WSL/Linux; ports exist for Windows Language: C Level: Beginner

Description: The classic Matrix effect in a terminal. The code is simple enough to hack colors, speed, and symbols.

Learn: Matrix effects, ncurses, timing, and screen redraws

Open code/repository
`https://github.com/abishekvashok/cmatrix`

34. TMatrix

Runs on: Modern terminal; Windows through WSL or compatible builds Language: Rust Level: Intermediate

Description: A more modern and accuracy-focused digital rain implementation. Good for studying performance and customization.

Learn: performant Matrix rain, terminal rendering, and configuration

Open code/repository
`https://github.com/M4444/TMatrix`

35. cxxmatrix

Runs on: ANSI terminal; WSL recommended on Windows Language: C++ Level: Intermediate

Description: Goes beyond Matrix rain: it includes scenes, banners, Game of Life, and Mandelbrot. A terminal visual lab.

Learn: Matrix rain, Conway, Mandelbrot, and alternating scenes

Open code/repository
`https://github.com/akinomyoga/cxxmatrix`

36. digi-rain

Runs on: Node.js/TypeScript in the terminal Language: TypeScript Level: Intermediate

Description: Good if you want to edit terminal animation using the Node ecosystem instead of C/C++.

Learn: modern digital rain effects, JavaScript CLIs, and incremental drawing

Open code/repository
`https://github.com/gfargo/digi-rain`

37. fakesteak

Runs on: C in a Unix/WSL terminal Language: C Level: Beginner

Description: A compact Matrix rain implementation. Better for reading the code than for flashy visuals.

Learn: simple Matrix screensavers, ANSI terminal control, and minimal distribution

Open code/repository
`https://github.com/domsson/fakesteak`

38. rusty-rain

Runs on: Rust in a cross-platform terminal Language: Rust Level: Intermediate

Description: Use it to learn Rust through simple animation instead of abstract, dull examples.

Learn: Rust CLIs, animation, render loops, characters, and emoji

Open code/repository
`https://github.com/cowboy8625/rusty-rain`

39. ASCII-renderer

Runs on: Linux/WSL; ncurses Language: C Level: Intermediate to advanced

Description: Renders 3D objects in the terminal using characters. An excellent bridge between linear algebra and retro visuals.

Learn: 3D ASCII rendering, cameras, lighting, and OBJ meshes

Open code/repository
`https://github.com/ShakedAp/ASCII-renderer`

40. ASCII Donut Animation

Runs on: C++ in CMD/PowerShell after compilation Language: C++ Level: Beginner to intermediate

Description: The famous rotating donut. If you want visual mathematics with a fast payoff, this one is mandatory.

Learn: 3D donuts, sine/cosine, simplified z-buffers, and ANSI colors

Open code/repository
`https://github.com/sherwinvishesh/ASCII-Donut-Animation`

41. AsciiCube

Runs on: Browser; editable locally Language: JavaScript Level: Beginner

Description: An ASCII cube in the browser. Good for studying geometry without a heavy toolchain.

Learn: 3D cubes, Bresenham, rotation, and projection

Open code/repository
`https://github.com/jehna/AsciiCube`

42. ascii-3d-cube

Runs on: PowerShell/CMD with Python Language: Python Level: Beginner

Description: A Python version of a 3D ASCII cube. Friendlier to edit than C++ if you want to learn by trial and error.

Learn: 3D cubes, gradients, depth, and interactive loops

Open code/repository
`https://github.com/msadeqsirjani/ascii-3d-cube`

43. ASCII-generator

Runs on: PowerShell/CMD with Python Language: Python / OpenCV Level: Intermediate

Description: A tool for turning media into ASCII. It can be adapted into animations, filters, and terminal demos.

Learn: image and video to ASCII conversion, OpenCV, and brightness mapping

Open code/repository
`https://github.com/vietnh1009/ASCII-generator`

4 - Terminal Games and Engines

44. FTXUI

Runs on: Windows/Linux/macOS terminal Language: C++ Level: Intermediate

Description: A C++ library for terminal interfaces. It is not a ready-made demo, but it is an excellent base for console games and visuals.

Learn: modern TUIs, layouts, events, and visual components

Open code/repository
`https://github.com/ArthurSonzogni/FTXUI`

45. ruscii

Runs on: Terminal; Rust Language: Rust Level: Intermediate

Description: An engine for making games in the terminal. Great for moving from passive animation into interactive work.

Learn: terminal graphics engines, input, sprites, and ASCII games

Open code/repository
https://github.com/lemunozm/ruscii

46. dotnet-console-games

Runs on: CMD/PowerShell with .NET Language: C# Level: Beginner to intermediate

Description: An educational collection from the .NET community. Very good for learning game loops without a graphics engine.

Learn: Snake, Pac-Man, 2048, Sudoku, Wordle, and console-game architecture

Open code/repository
https://github.com/dotnet/dotnet-console-games

47. CSharpConsoleGames

Runs on: CMD/PowerShell with .NET Language: C# Level: Beginner

Description: Classic games in C# for the console. Simple, editable, and useful for understanding state, collision, and input.

Learn: Tetris, Snake, Tron, Pong, cars, and console drawing

Open code/repository
https://github.com/NikolayIT/CSharpConsoleGames

48. cli-games

Runs on: Node.js in the terminal Language: JavaScript Level: Beginner

Description: A collection of terminal games in JavaScript. Good for rapid prototyping in CMD/PowerShell with Node.

Learn: simple CLI games, menus, input, and game logic

Open code/repository
https://github.com/Seniru/cli-games

49. tetrigo

Runs on: Terminal with Go Language: Go Level: Intermediate

Description: Tetris in the terminal written in Go. Good if you want to learn a more modern CLI architecture.

Learn: Tetris, Bubble Tea/TUI, input, and render loops

Open code/repository
https://github.com/Broderick-Westrope/tetrigo

50. samtay/tetris

Runs on: Terminal; Windows through source/WSL Language: Haskell Level: Intermediate to advanced

Description: Tetris in Haskell. Not the easy path, but it teaches a different way to think about state and rendering.

Learn: functional Tetris, Brick/Vty TUI, and functional architecture

Open code/repository
https://github.com/samtay/tetris

51. pokete

Runs on: Terminal; Python Language: Python Level: Intermediate

Description: A large terminal game. Good for studying how an ASCII project grows from toy into real game.

Learn: Pokemon-style RPGs in the terminal, maps, ASCII sprites, and state handling

Open code/repository
https://github.com/lxgr-linux/pokete

52. csol

Runs on: Terminal / DOS / ncurses or pdcurses Language: C Level: Intermediate

Description: A solitaire collection for the terminal. A good example of clean TUI design and traditional games without modern graphics.

Learn: terminal solitaire, text interfaces, and portable input

Open code/repository
https://github.com/nielssp/csol

5 - DOSBox, x86, VGA, and Retrocoding

53. masm-tasm

Runs on: VS Code + DOSBox/JSDOS/msdos-player Language: Assembly 8086 Level: Beginner to intermediate

Description: One of the most practical entry points for DOS Assembly today. It sets up the edit-compile-run cycle.

Learn: running and debugging MASM/TASM, and a modern 8086 workflow

Open code/repository
https://github.com/dosasm/masm-tasm

54. dos-dev-template

Runs on: VS Code + DJGPP + DOSBox-X Language: C / C++ Level: Intermediate

Description: An excellent template for creating DOS apps or games with modern tooling and sane debugging. Far more rational than suffering in the dark.

Learn: modern DOS templates, remote GDB, mode 13h, and asset handling

Open code/repository
https://github.com/badlogic/dos-dev-template

55. dos3d

Runs on: DOSBox Language: C Level: Intermediate to advanced

Description: 3D rendering in DOS. If your goal is to understand low-level graphics, this is well aligned.

Learn: software rendering for DOS, mode 13h, and Doom/Quake-style foundations

Open code/repository
https://github.com/kondrak/dos3d

56. 54-byte Snake

Runs on: DOSBox / DOS Language: x86 Assembly Level: Advanced

Description: Snake in 54 bytes. Absurd, but excellent for understanding the cruel and elegant spirit of sizecoding.

Learn: extreme sizecoding, tiny .COM files, and games in almost no bytes

Open code/repository
https://github.com/donno2048/snake

57. Tetris Assembly 8086

Runs on: EMU8086 for compiling; DOSBox for running Language: Assembly 8086 Level: Intermediate

Description: Tetris in Assembly with instructions for running in DOSBox. Better than isolated examples because it is a complete game.

Learn: complete 8086 games, screen handling, input, and piece logic

Open code/repository
https://github.com/mkh2097/Tetris-Assembly-8086

58. Joe Wing DOS Programs

Runs on: NASM + DOSBox Language: 386 Assembly Level: Intermediate

Description: A collection of DOS Assembly games with available source code. Excellent for retrocoding without taking on a giant project.

Learn: small DOS games, Tetris, Lights Out, puzzles, and assembling with NASM

Open code/repository
https://joewing.net/projects/dos/

59. Wolfenstein 3D source

Runs on: Borland C++/old environment; source-code study Language: C / x86 / DOS Level: Advanced

Description: The original Wolfenstein 3D source code. It is historic and educational, but it is not a modern plug-and-play project.

Learn: raycasting, VGA, game loops, and old FPS engine architecture

Open code/repository
https://github.com/id-Software/wolf3d

60. DOOM Open Source Release

Runs on: Original Linux build; modern ports for Windows Language: C Level: Advanced

Description: Doom source code is mandatory engine reading. Warning: commercial assets are not included, and the official release is not the original DOS version.

Learn: BSP, Doom rendering, engine architecture, and separated assets

Open code/repository
https://github.com/id-Software/DOOM

61. DOS Games Archive - Source Code Games

Runs on: DOSBox Language: C / Pascal / Assembly / varied Level: Varied

Description: A page of DOS games that include source code. Use it for mining examples, always checking license, age, and build instructions.

Learn: DOS games with source code and comparisons between old coding styles

Open code/repository
https://www.dosgamesarchive.com/license/source-code

62. DOS Haven Sources

Runs on: DOSBox / FreeDOS Language: C / Pascal / ASM Level: Beginner to advanced

Description: A collection of links to DOS source code and tutorials. Good for building your retrocoding curriculum.

Learn: VGA, mode 13h, tutorials, and DOS programming examples

Open code/repository
https://www.doshaven.eu/sources/

6 - Emulators and Useful Bridges

63. js-dos

Runs on: Browser / JavaScript Language: TypeScript / C++ through DOSBox Level: Intermediate

Description: Useful when you want to turn a DOS demo or game into something that runs in the browser.

Learn: running DOS programs in the browser and packaging retro experiences

Open code/repository
https://github.com/caiiiycuk/js-dos

64. DOSBox Staging

Runs on: Windows/Linux/macOS Language: C++ Level: Intermediate

Description: A modern emulator for running and testing DOS demos and games. It is a base tool for the DOSBox part of this track.

Learn: DOS emulation, audio, video, configuration, and compatibility

Open code/repository
`https://github.com/dosbox-staging/dosbox-staging`

65. 1oom

Runs on: Terminal/SDL; requires original game data Language: C Level: Advanced

Description: A recreation of the classic Master of Orion engine. Good for studying portability and reimplementation of older systems.

Learn: engine recreation, turn-based strategy, and retro compatibility

Open code/repository
`https://github.com/1oom-fork/1oom`

Consulted Sources and Validation Points

This list was built from repository pages, GitHub topics, demoscene lists, and DOS source-code archives. Still, before investing hours in a project, check its license, build instructions, maintenance date, and open issues.

- Awesome Demoscene: https://github.com/psykon/awesome-demoscene
- Teach Yourself Demoscene in 14 Days: https://github.com/psenough/teach_yourself_demoscene_in_14_days
- Demoscene Starter Kits: https://github.com/anttihirvonen/demoscene-starter-kits
- Second Reality source: https://github.com/mtuomi/SecondReality
- GitHub topic: ascii-animation: https://github.com/topics/ascii-animation
- GitHub topic: terminal-animation: https://github.com/topics/terminal-animation
- GitHub topic: dosbox: https://github.com/topics/dosbox
- DOS Games Archive - source-code games: https://www.dosgamesarchive.com/license/source-code
- DOS Haven - sources: https://www.doshaven.eu/sources/
- Bonzomatic: https://github.com/Gargaj/Bonzomatic
- Farbrausch fr_public: https://github.com/farbrausch/fr_public

Practical Next Step

Choose three projects: one PowerShell/CMD project, one ASCII 3D project, and one DOSBox project. Make one visible modification in each: change the palette, accelerate the loop, replace the characters, change input, or add a credits screen. That is worth more than opening fifty repositories and compiling none of them.