**NeptunianAxolotl Ludum Dare Skeleton Framework**

**Technical Manual**

## Handlers

**resourceHandler.lua**   
Introduced in LD45 (Warp Trash Flotilla)

Loads all images, handles animations, isometric animations, facing, alpha, rotation, scaling, etc.. It has barely changed since LD47, and I don't know if it was touched at all in LD49 and LD50. So I'd call it the most stable system. Loads from resources/.

**chatHandler.lua**   
Introduced in LD45 (Warp Trash Flotilla)

writes a console to the screen. Extended in LD49 to delay timeouts until abstract "turns". Now line wraps correctly. The way it draws is reworked each LD as requirements for things like scrolling up or down changes. Could be a bit more general.

**soundHandler.lua**   
Introduced in LD45 (Warp Trash Flotilla)

Handles sound loading and playing. Sound tends to be a bit nonsense due to its lateness, so we have fought soundHandler for a few LDs. I went in and fixed some things last week, so maybe we're done fighting. It's hard to say when it was really extended beyond what it originally did, rather, the bumps have been slowly pushed around and ironed out.

**musicHandler.lua**  
Not sure when introduced.

Tells soundHandler to play music. Something like this has existed since LD45, but in various rewritten forms as requirements shifts, were dropped, or bumps in soundHandler caused issues. Did tracked in LD45, didn't do much until LD49. I fixed lingering issues last week, but I can imagine music coming along that wants it to be extended again.

**effectsHandler.lua**   
Introduced in LD47 (Crocodial)

A particle system that sees use if we have time. Can draw floating text for LD48, explosions, does gravity, etc. Can draw on UI or in-world. Would have been used for smoke and sparkles this time. Reads from effects/, and has a generic effect spawing API.

**physicsHandler.lua**   
Introduced in LD49 (Three Miles High)

A simple wrapper that sets up and runs love's physics engine without cluttering other files. Handles callbacks for collisions. Barely needs thinking about.

**deckHandler.lua**   
Introduced in LD48 (Polyomino Prospector)

A simple system that maintains a global deck to which cards can be added or drawn. Many things can be modelled as decks under the hood (LD49 runs on deck handler). Should probably be an object.

- dialogueHandler.lua (LD50). A system that handles branching, conditional, and dynamic dialogue trees. Takes generic 'chat' "objects" in which contain setup information and a map of scenes in a particular format. Fairly general, but only handles one chat at a time (no parallel or saving chats). Uses chatHandler to draw the output. I'm very happy with my early decision to keep chatHandler and dialogueHandler separate.

- global.lua (LD???). A file containing a table of globals. I've been building up a best practise of adding to this file whenever I write a number which I \_might\_ want to tweak in the future. Perhaps more importantly, anything toggled for debug purposes should have a toggle in globals, rather than via comments. Commenting and uncommenting code for debug is a recipe for leaving things in a broken state.

- progression.lua (LD47). A generic system for specifying how to turn a single number, "progress", into distributions of game elements that the player encounters. Specifies probability distributions over time using keyframes. Enemies and terrain from LD48 and LD49 run on this.

- shadowHandler.lua (LD50). Wrapper for a shadow framework I downloaded a few years ago but never used in a LD. I had no experience with it and had to fight it a few times when it tried to wrestle away our framerate. Also it really wants to be used in top-down games. Making it do isometric somewhat passably took a few tricks. Somewhat specialised to LD50, and isometric view.

- componentHandler.lua (LD49). Not really a handler. Specialised to LD49 then cut down to demonstrate physics.

- gameHandler.lua (LD??). Not really a handler either. Exists as a prompt to collect all the "gamey" stuff in one place rather than in world objects. LD50 doesn't really have global "gamey" stuff though, such as money or score.

- moduleTest.lua (LD??). Draws a walking monk, animations, explosions, sounds. Now has shadows. Test for the generic handlers.

- groundHandler.lua (scratch) - A new version of this type of file exists in various games as required. This one keeps track of the ground tiles and draws them. Importantly contains transforms TileToPos and PosToTile. These transforms break if the ground is given odd offests.

- npcHandler.lua (scratch) - A good example of a simple skeleton that tracks and runs an Object (in the OO sense). These appear in games as required. This one runs all the NPC fireflies.

- terrainHandler.lua (scratch) - A more complicated example of the same task as npcHandler. This one handles terrain features (which includes structures). Handles things like checking feature placement and finding features within a distance, or some other filter. The form of this type of file is determined by the game and built with other blocks. Some of its functions could be moved to util though.

- playerHandler.lua (scratch) - A file that tends to exist and balloon a bit out of control. Handles player UI, player input, and a few things about player state.

- main.lua (LD45?) - Entry point. Kept slim by calling world. If the interface of a game is complex there might also be interface.lua.

- world.lua (LD45?) - The primary entry point on from main.lua. Initialises and runs most handlers. Contains generic things such as music state and whether the game is over. Contains transforms between screen, world, and interface (eg WorldToScreen, ScreenToInterface). I set the interface->screen transform to the identity in this game for simplicity though. The game is restarted by setting world.self = {} and calling world.Initialize.

Handler practises: Most handler-style files now contain self and api local tables. Functions for export go in api, and local data goes in self. Self should be cleared in initialize. This system confines all gamestate variables to limited domains, so makes restarting the game very easy. Restart was supported from the start and only had one bug. Restarting so easily with love is not a forgone conclusion.

common utilities:

- utilities/cameraUtilities.lua - Handles smooth camera movement, updating to keep specified circles in view, that sort of thing. Sets camera transform used by world.

- utilities/delay.lua - Adds Delay.Add for making functions happen at some later time.

- include/font.lua - Runs fonts. Mainly forces people to set a font scale (eg heading 1, heading 2) rather than make up a font size when they need it. Keeps things looking consistent.

- include/IterableMap.lua - Data structure for seamless object adding/removal and foreach-like functionality. If you're frequently changing the length of a list, or doing pairs on a complicated map, you probably want iterableMap.

- include/PriorityQueue.lua - A priority queue data structure. Mainly used so to register then draw functions back-to-front for isometric games.

- include/util.lua - A collection of vector, probability, and data handling utilities that can turn a dozen lines of code into one. Basic vector operations, transforms, distances between points and lines, polar coordinate conversion, hit tests, sampling from discrete probability distributions, sampling cardinal directions, uniform sampling from things like circles and ellipses, smooth stepping, timer updating, table printing, table deepcopy.

On vectors: I decided that vectors are to be denoted {x, y}. This keeps the utilities nice and chainable and cuts down on extra variables in code.