

Design Philosophy

In an earlier epoch I designed embedded system hardware, software, and development tools in two different areas:

- Pro Audio
- Digital Signal Processing

Both of these fields are 'real-time' programming. It's not dissimilar to what you have to deal with in Unity3D:

- A frame in pro audio at 44.1 kHz doesn't allow much time per-sample to process much, about 22 microseconds.
- A frame in a game at 60 Hz is about 17 milliseconds which sounds quite generous by comparison.

Either way, you have a certain amount of time to do your processing or:

- Audio ==> Audible 'artifacts' such as clicks and pops
- Game ==> Dropped frames

This is all to say that this biased my approach to coding: performance and memory are #1.

So I tend to eschew the modern programming idiom of write first optimize later.

- Use inheritance and interfaces for Tile classes.
- Use generics where it makes sense (to me), such as in the `Messaging` Service's messages.
- Tightly hardcode things that by nature have to be optimized, such as the `Tweener` Service and the internals of `TpLibTasks`.
- Extensively pool class instances and other objects.
- Extensively cache tile instances in the layout system.

No acronym-based coding style will be found anywhere :-)

Revision #11

Created 14 July 2025 11:41:36 by Vonchor

Updated 16 July 2025 15:06:48 by Vonchor