Devlog 7 - December, 20, 2025. Cold Feet

I've been playing a lot of Rogue Trader this past week, which has made development slow down somewhat. I think last weekend were the first two consecutive days I hadn't worked on the game at all ever since I started. It gave me the idea of implementing a notifications tray which could use our new tooltip manager, but that's hardly a positive trade-off overall.
Another thing that has slowed me down is the fact that I'm very much dragging my feet on the new release. So I'm both not willing to work on big stuff to not let it half-broken by the time of the release, and also not feeling like working on the more administrative tasks that would improve the release.

Part of this is updating the 'Help' topics, which I hadn't touched ever since introducing them I think almost 3 months ago now. Won't be able to get everything done, but it's easy enough to put up a disclaimer. I will also mark which version they are updated last, which will make further updates easier, since it will be apparent what has been changed since.
I also tried adding more names using the new nationality-ethnicity structure, which I've been doing mostly manually and has led to reconsidering the whole thing. That type of structure is probably fine for 'real' names, and also easier to automatize, but for aliases it just becomes a massive clutter of repeated pools, because most 'alias tokens' are used by performers from all over and of every ethnicity. I think a 'global' pool plus a few ethnic ones would probably be enough, and there should also be a way to tag some pools with 'physical' and 'behavioural'? alias tokens (like Petite, Mistress, Squirt...). I haven't properly thought about it yet, so I've put everything on hold for now.

What Was Done

The first addition for this cycle was the monthly system I talked about in the last devlog. I've settled on 13 months for now, each with 4 weeks (so still 52 weeks per year). This has led the date formatting is a bit all over the place at the moment: the main view is MM/W/YYYY, the new Roster view uses YYYY/MM/WW for contract start and end dates, the Schedule views (tab and widget in Talent Profile) use weeks, as does the inbox...
Aside from being used by the AI studios to 'decide' how many and when to release new scenes, it is also used for the exclusive contract scene system limit. Which also checks if there are partial months either at the beginning or the end of the contract and performs a calculation to determine how many scenes those should translate to.

Additionally, the AI studios now have archetypes. Similar to the Talent ones, it determines the focus targets, orientations, and tags they will use in their scenes. They now create and release scenes with tags taken from that pool that get a random quality. These scenes now follow the same revenue process and market effects as the ones the player releases.

This dual usage necessitated the streamlining of the revenue calculation process: it used to take a full Scene object, which the AI didn't produce (no cast yet, for example), so the AI would have to mock this whole object to make use of that calculation. Instead, now the revenue calculator takes a DTO (talked about how I started using those back in the service refactor, I think in Devlog #2) which only carries the values from a Scene that the revenue calculation needs, which the AI studios all 'simulate' already.
This means that the AI scenes now generate a revenue depending on the market interest created (this revenue isn't added to the Studios total money at the moment, just because), which also lowers the Spending Willingness of the viewer groups affected, which at the moment means that the more affected viewer groups usually get spent in a couple of months and then stay at a very low rate for the rest of the game. Obviously needs a balancing pass, but it's not a priority at the moment.

Another inclusion that was on-hold previously is the tag builder for the ethnic tags. At the moment, it includes sub-groups (East Asian, Western European, etc) for the individual ones and only the main ethnicities for the interracial/same-ethnicity pairs (I decided to split them in the end). I guess it would make sense to have all the sub-ethnicities included there as well, but there were already a lot of them just with the main ones, so that's how it is for now.
This is somewhat alleviated by the addition of the collapsible categories view, which I only added to the Physical pane of the Planner, which actually doesn't separate tags by category, but by concept and orientation.

I also created an extra value for the Action Segments data structure: its role (giver/receiver/performer). Previously, since it wasn't explicitly provided, all the UI and logic consumers of said data had to perform a pretty wobbly operation to extract the role string from the segment object, which involved cutting up strings and whatnot, now it simply calls the object's role value, so we've turned 10 lines of code to only 1. This doesn't have any impact on what the player sees, but it makes the backend more understandable and robust.

While testing this, I discovered a couple of issues: Chemistry creation checked values already in the database to see if it should create a new one or not, but it didn't check for 'discoveries' still uncommitted (i.e. that had occurred this same week/turn), which would make the final commit for chemistry add several entries for the same relationship, which throw an error. So now it also checks still uncommitted additions.
The second issue had to do with the recently created action tag builder, particularly the more complex logic for on-the-fly Straight tag definition: for some tags, like Anal, where the receiver can be Male or Female, it would check the performer's gender once added to either role, and then make the other role deny performers of that gender. Except that this is not what it did: it simply disallowed any other performer for being added to the segment, which wasn't apparent in tags with only one receiver and one giver, but became apparent as

soon as more performers were involved. Now it works as it should.

Also refactored the email handling, as their content was hardcoded in the logic layer. To move it to the data layer I'm using Jinja2, which allows the use of variables, needed in emails as opposed to something like the help topics. So now the email service has convenience methods to easily map variables for the emails that use it (one for market discovery and one for talent going on tour, so far), and the ones that don't (welcome email, for now) just use its main method directly.

The code review I talked about doing some devlogs ago after every new feature implementation or refactor led to fixing an issue with the talent details tooltip persisting on-screen even when the application window 'lost focus' (like when it was Alt-tabbed from), which I had been trying to fix for over an hour back when it was introduced and couldn't in the end. Since I had marked it with a 'FIXME', the AI picked it up and provided a working fix on the first try, which I was very sceptical about. A good incentive to keep doing these reviews as well as being more thorough with the annotation tags.

Finally, I created a Roster view to show all the contracted talent, which I hadn't done back when I implemented the exclusive contracts. It's yet another table, this one showing all the terms of the contract. At some point it might have to be something prettier if the contract is still working as it does now, because showing all the available orientations and concepts isn't very proper on a table. In any case, I have to standardize tables and how they interact with the button to select the visible columns, and how those are stored, because none of the many tables in the game work exactly the same.
This view is the first one that uses the newly-created base classes for modeless views and presenters, both incorporating all the logic that the combo uses throughout the game: integration of the geometry mixin for saving and loading of window position and size, dialog 'windowfication' for the better layout behaviour, etc. The presenter has a cleanup method that will hopefully prevent anymore cases of zombie presenters, by explicitly cleaning every signal connection via a UI Manager call once the view is closed. Now I have to make every modeless dialog/presenter combo in the game use these, which hopefully won't be too much trouble.

Public Release

The closer the deadline for the more public 'announcement' of the game gets, the more doubts I have about it.
The project is not properly set-up for collaboration: documentation is spotty at best and I don't yet even have a README in GitHub. The whole codebase can't be in great shape by how it has come to be and who's been developing it, but I have no idea about how bad it actually is, and it might be worse than I believe it is.

The game is also in a very rough state, with not clear direction in either gameplay or UI design. It feels like I'm trying to find people that will turn a pile of trash into something.
I guess I'm just very insecure about everything and I'm afraid of exposing it to a wider public. Realistically, the worst that can happen is that it is completely ignored, and I basically already have that anyway. I won't be creating a Discord server for the moment tho, because seeing it dead might make me too depressed to keep going with the development lol.
In any case, I'm going to be setting up the development thread and releasing the build before Christmas just as I had planned, and I'll probably take a couple of days off (hopefully the build isn't too broken or no one tries it).