OpenVic

Dev Diary 8

Salutations!

Welcome back for our 8th OpenVic Dev Diary! It's been a while since our last instalment, so we'll begin with an overview of what the project has been up to in the interim;

- With the completion of a major development phase, a significant portion of the game's dataloading is now complete. Thanks to this we have also begun working on the game's simulation, alongside further additions to the UI.
- We constructed a firm framework upon which we could build the game's data and mechanics, which involved reorganising old code and preparing space for brand new code.
- From there, we started on the main simulation, which is full of equations and algorithms. This facilitates functions such as calculating country scores, buying and selling goods, scripts for decisions and events, and all sorts of other stuff that makes up the game logic.
- To ensure that we are perfectly emulating the behaviour of Victoria 2's systems, research spearheaded by our dedicated team members such as General WVPM, alongside a wide collection of modders and Vic2 experts have diligently combed through over a decade's worth of guides, forum posts and occasionally-accurate wiki entries, all with the aim of verifying all these claims ourselves and ensuring a standard for accuracy and faithfulness to Vic2
- We also created several special "research mods", which contain a minimal set of defines, allowing us to test specific game mechanics in a clean environment.

All this work has allowed us valuable insight into the limits of Vic2's defines parsing, demonstrating exactly what the game requires and how it behaves with unexpected inputs, which allows us to emulate Vic2's mod handling as closely as possible. For equations, we have been documenting our confirmed findings in separate text files to the code itself (hopefully more legible to non-developers) which should hopefully assist all hoping to further understand how Vic2 actually works.

SIMULATION:

ARCHITECTURAL WORK:

With dataloading mostly solved, our focus has shifted towards working on the game simulation. During this we have decided to make some architectural changes to the game's core code. One of the major changes has been separating everything into two separate sections, Definition and Instance.

The Definition section contains the constant data loaded by the game during startup, which should not change after that point. The Instance section, as you might guess, contains the opposite, made up of mutable game-state data, which changes over the course of a game session.

This approach allows us to easily start new game sessions, as we can simply destroy the instance section and re-initialize the entire list, instead of resetting every potential variable. Additionally, this prevents accidental modification of constant game data after it has been loaded.

There are also additional technical changes, such as making use of various specialised containers, tailored to best fit the needs of specific types of data making up the game-state. Some systems, such as the handler for the contents of defines.lua, have been completely rewritten to improve performance. Mapmodes have also been reworked to better fit the vanilla mapmodes that it will primarily use in game. Some minor gaps in the dataloader system have been patched, such as the loading sound effect and song chance defines, alongside a variety of UI defines.

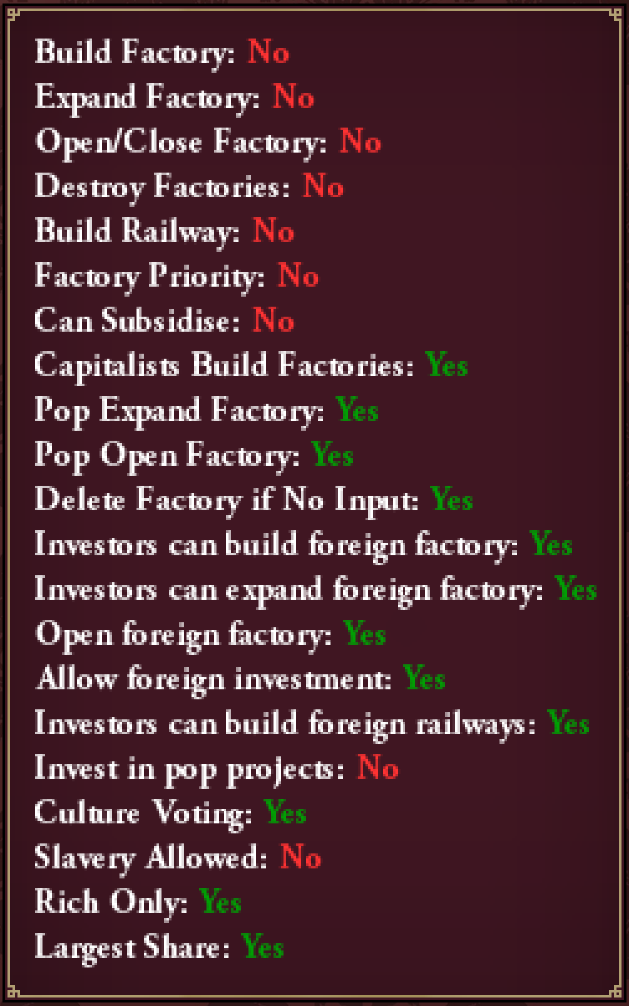
Modifiers and Rules:

Many of Victoria 2's mechanics are affected by modifiers, meaning that these are an essential component to OpenVic.

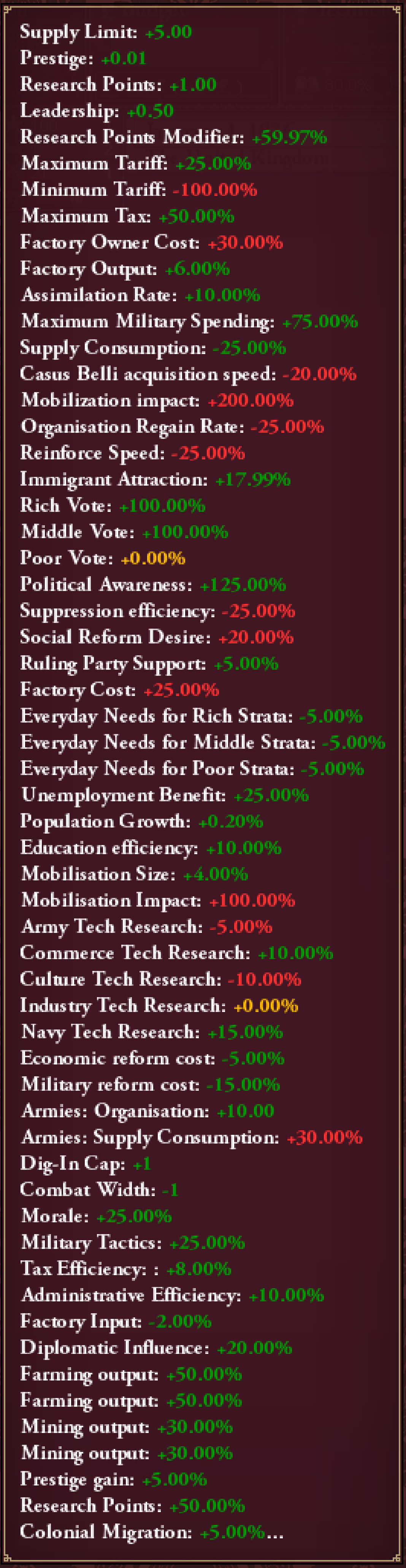
The modifier system contains a multitude of components; Modifiers, ModifierEffects , ModifierValues, ModifierSums, the ModifierManager and the various caches for ModifierEffects that it contains.

Modifiers are broadly split in two sections, with one side being defined modifiers that can be applied to things and the modifier effects that form them, and the other side being tracked modifiers and the sum of their effects on a particular object. The objects that modifiers are applied to are countries, provinces and military units, with much of our efforts being focused on countries and provinces so far.

On every game-state update, we sum up all the modifiers that are currently affecting each country and province, alongside the sum of each province's owner and further the sum of each country's own provinces to get the final sum of modifiers for each country and province.



Build Factory:	No
Expand Factory:	No
Open/Close Factory:	No
Destroy Factories:	No
Build Railway:	No
Factory Priority:	No
Can Subsidise:	No
Capitalists Build Factories:	Yes
Pop Expand Factory:	Yes
Pop Open Factory:	Yes
Delete Factory if No Input:	Yes
Investors can build foreign factory:	Yes
Investors can expand foreign factory:	Yes
Open foreign factory:	Yes
Allow foreign investment:	Yes
Investors can build foreign railways:	Yes
Invest in pop projects:	No
Culture Voting:	Yes
Slavery Allowed:	No
Rich Only:	Yes
Largest Share:	Yes



Supply Limit: +5.00
Prestige: +0.01
Research Points: +1.00
Leadership: +0.50
Research Points Modifier: +59.97%
Maximum Tariff: +25.00%
Minimum Tariff: -100.00%
Maximum Tax: +50.00%
Factory Owner Cost: +30.00%
Factory Output: +6.00%
Assimilation Rate: +10.00%
Maximum Military Spending: +75.00%
Supply Consumption: -25.00%
Casus Belli acquisition speed: -20.00%
Mobilization impact: +200.00%
Organisation Regain Rate: -25.00%
Reinforce Speed: -25.00%
Immigrant Attraction: +17.99%
Rich Vote: +100.00%
Middle Vote: +100.00%
Poor Vote: +0.00%
Political Awareness: +125.00%
Suppression efficiency: -25.00%
Social Reform Desire: +20.00%
Ruling Party Support: +5.00%
Factory Cost: +25.00%
Everyday Needs for Rich Strata: -5.00%
Everyday Needs for Middle Strata: -5.00%
Everyday Needs for Poor Strata: -5.00%
Unemployment Benefit: +25.00%
Population Growth: +0.20%
Education efficiency: +10.00%
Mobilisation Size: +4.00%
Mobilisation Impact: +100.00%
Army Tech Research: -5.00%
Commerce Tech Research: +10.00%
Culture Tech Research: -10.00%
Industry Tech Research: +0.00%
Navy Tech Research: +15.00%
Economic reform cost: -5.00%
Military reform cost: -15.00%
Armies: Organisation: +10.00
Armies: Supply Consumption: +30.00%
Dig-In Cap: +1
Combat Width: -1
Morale: +25.00%
Military Tactics: +25.00%
Tax Efficiency: : +8.00%
Administrative Efficiency: +10.00%
Factory Input: -2.00%
Diplomatic Influence: +20.00%
Farming output: +50.00%
Farming output: +50.00%
Mining output: +30.00%
Mining output: +30.00%
Prestige gain: +5.00%
Research Points: +50.00%
Colonial Migration: +5.00%...

We are careful not to double-count any modifiers, for example an owned province contributes modifiers to its owner and receives its owner's contribution (this includes the contributions from other owned provinces), while ensuring that it does not receive copies of the modifiers it contributed to the country itself.

We have tried two distinct approaches to implementing this system; one where the countries contribution is added to their owned provinces as a part of the game-state update alongside the countries receiving their owned provinces' contribution, and the other where only the province contributions are fully added to their owner countries while the country contributions to their provinces are kept separate and only added for specific modifier effects at the moment they're requested to be used elsewhere in the game code.

In the end, both of these approaches were implemented, with a toggle within the game code controlling which is used during compilation. This allows us to compare the performance of these approaches during development and make heavier use of the modifier system, as well as compare the output results to more easily spot bugs.

Rules, while similar to Modifiers, are not as thoroughly tied into the game mechanics. Due to the similarities of these systems, we integrated them alongside each other. Rules are only applied at the country level, and even then there are only a handful of sources.

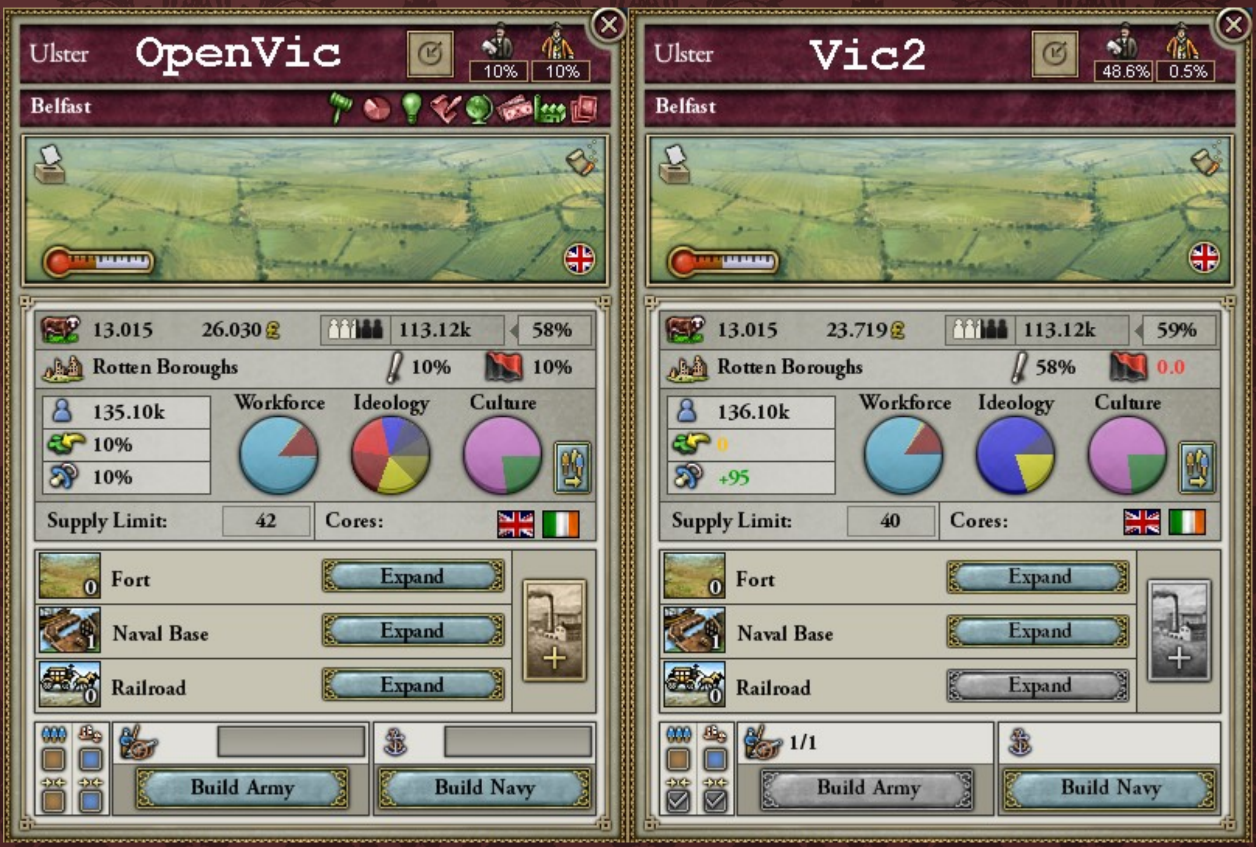
Every game-state update we calculate the overall set of rules affecting each country, allowing them to be queried elsewhere in the code to control whether certain actions are allowed.

Economy:

RGOs:

Each land province can have a Resource Gathering Operation (RGO). An RGO has no inputs and produces a single output good. RGOs are assigned to provinces in the history defines file (for example: “trade_goods = timber”). The trade_goods have to be converted into a production type. OpenVic now supports this and also helps modders validate their history by providing warnings for invalid goods.

Vic2 vanilla uses two types of production templates: ”RGO_template_farmers” and “RGO_template_labourers”, each with their own worker POP type. Farmers and labourers are defined as equivalent to each other, meaning that when a province changes its RGO production type, equivalent pops are converted so that they can be employed by the RGO. This is supported in OpenVic just like Vic2.


Ulster OpenVic
10% 10%
Belfast
13.015 26.030 113.12k 58%
Rotten Boroughs 10% 10%
135.10k
10%
10%
Workforce Ideology Culture
Supply Limit: 42 Cores:
Fort Expand
Naval Base Expand
Railroad Expand
Build Army Build Navy

Ulster Vic2
48.6% 0.5%
Belfast
13.015 23.719 113.12k 59%
Rotten Boroughs 58% 0.0
136.10k
0
+95
Workforce Ideology Culture
Supply Limit: 40 Cores:
Fort Expand
Naval Base Expand
Railroad Expand
1/1
Build Army Build Navy

Each province has a province size determined when starting a game or loading a savegame. The maximum workforce size of an RGO scales with the province size. This size calculation has been known for years, although it took us a little while to determine which sources of rgo_size modifiers are included in the calculation, but OpenVic should emulate Vic2 perfectly here.

At the start of each playthrough there is a procedure that kickstarts the economy; workers are hired, goods are produced and taxes are collected. We haven't figured out exactly how this works yet. For now each RGO just hires all available workers.

Not all workers work the same though. Some workers simply provide throughput, others modify output efficiency. You can even create a worker that modifies input efficiency. We figured out how these effects combine with the effects from modifiers and implemented it in OpenVic.

So we have the trade_goods to production type mapping, the pop conversion to equivalents, province size, hiring of workers and the modifiers. That's all we need to have RGOs produce goods. All this means that RGOs in OpenVic now produce goods, sell them, and pay income to POPS.

Market:

POPs often have greater needs than can be satiated by local RGO produce, chiefly being consumer goods. Fortunately, thanks to it being the 19th century, they can use currency to buy these goods. That however leaves a question; How much are these goods worth?

This requires a market. We discussed ideas for how this might work, during which General WVPM created a demo in Python. Modders may wish to add their own mechanics to the market, and Vic2 has its own base mechanics. As a result, the OpenVic market implementation is designed to allow the changing of how goods are traded and the way prices for those goods are determined.

The system is designed as follows: Every actor (be it a POP, RGO, factory or country) can place buy or sell orders into a central market system. The market then determines the price for each good and informs each actor how much was bought/sold and how much money is gained or was left after buying.

For buying, we created a custom order called the "Buy-up-to order". The buyer states the desired quantity and the maximum total cost. If the price multiplied by the desired quantity exceeds the maximum total cost, the buyer is forced to buy less. If the same calculation does not exceed the maximum total cost, (and assuming that there is sufficient supply of the good) the desired quantity is bought and the leftover money is returned.

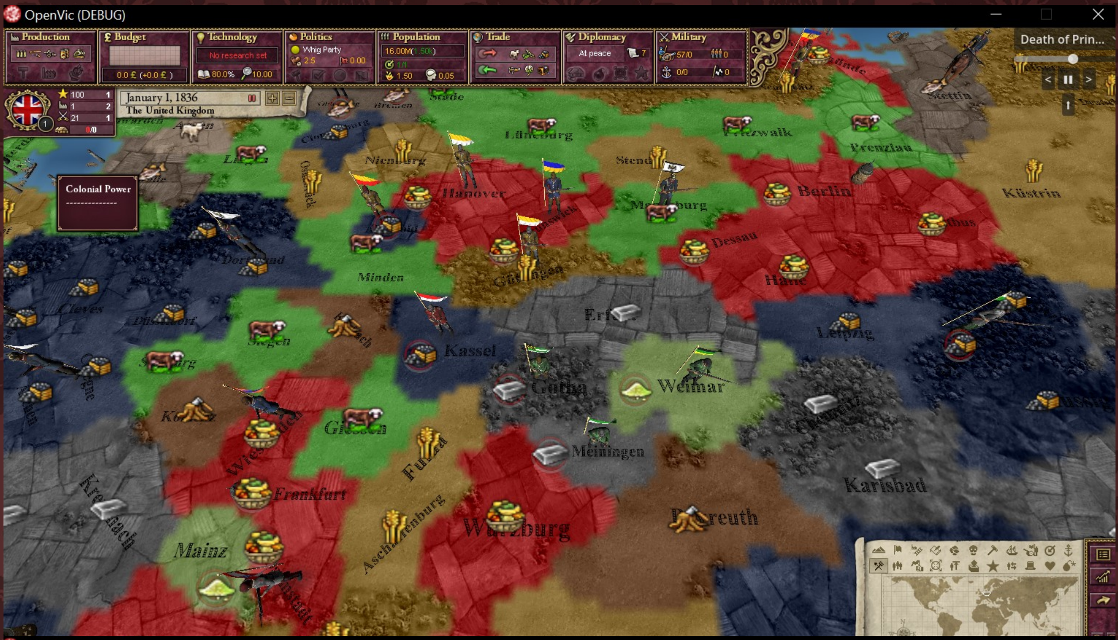
Buyers allocate their money when placing their orders, which means that this allocation has to be optimal. If for example an artisan producing steel requires 5 coal and 20 iron, the optimal allocation of money would result in an equal portion of coal inputs and iron inputs. This means that buying 3 coal and 12 iron is preferred over buying 2 coal and 20 iron instead.

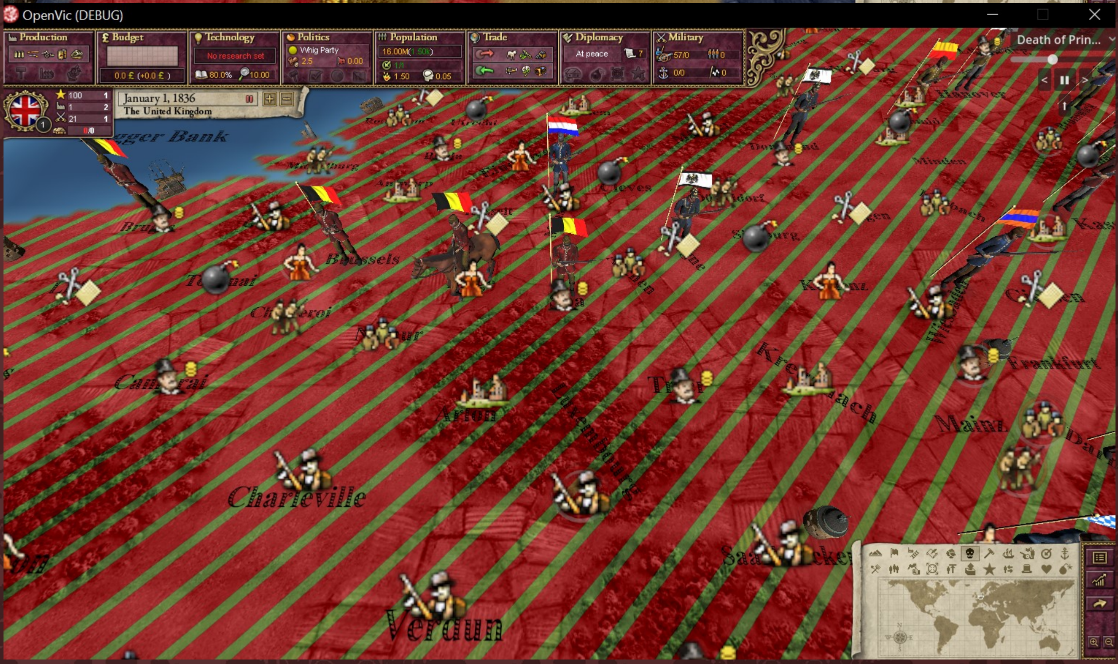
For selling, we currently only support one type of order, the market sell order. The seller states his intention to sell a certain quantity of a good for whatever price he can get. This does not guarantee that all goods are sold, however.

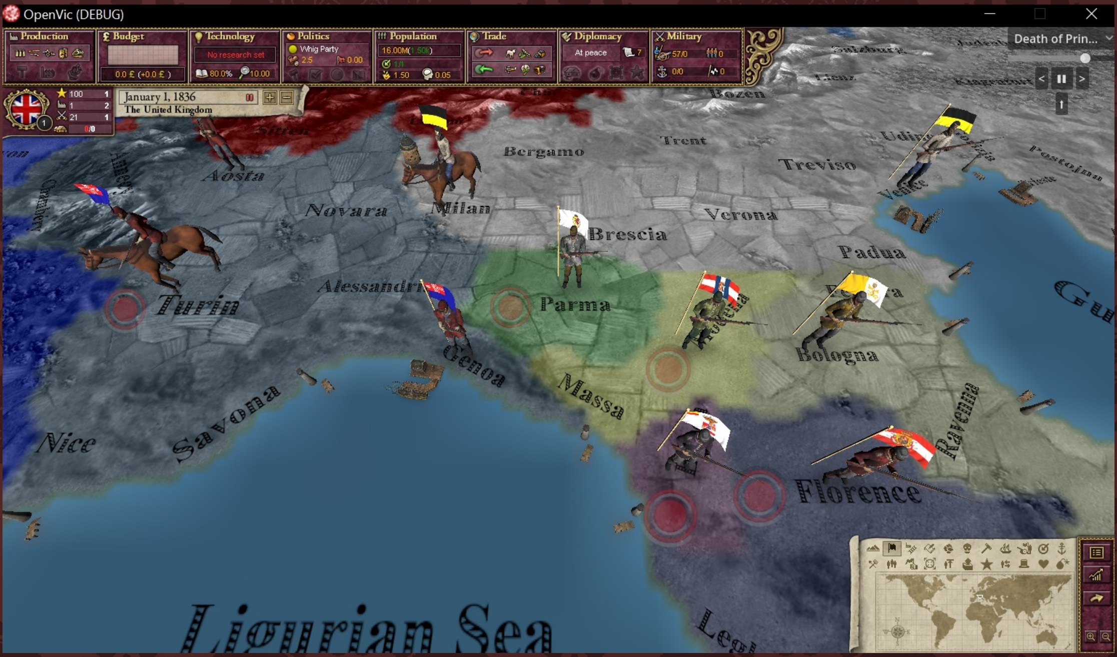
<u>UI</u>

In the latest version of OpenVic you can see that we are now using Vic2's minimap and map menu. We've also made significant jumps with other map modes since the last dev diary, with many of these map modes now working much closer to expectations now.

Take a look at the RGO map mode:

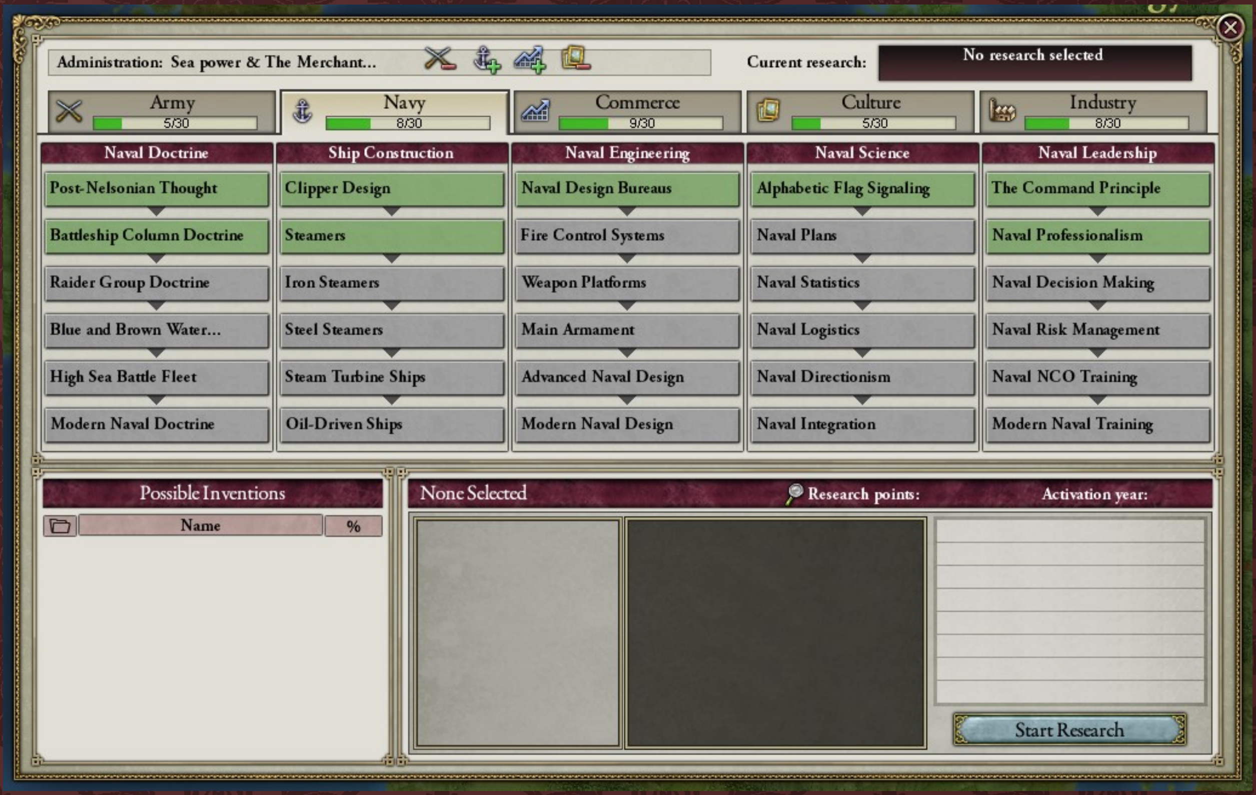

OpenVic (DEBUG)
Production
Budget
Technology
Politics
Population
Trade
Diplomacy
Military
No research set
Whig Party
At peace
January 1, 1836
The United Kingdom
Colonial Power
Death of Prin...
Hanover
Berlin
Minden
Kassel
Gotha
Weimar
Leipzig
Frankfurt
Mainz
Meiningen
Würzburg
Bayreuth
Karlsbad
Küstrin
Dessau
Erfurt
Fulda


OpenVic (DEBUG)
Production
Budget
Technology
Politics
Population
Trade
Diplomacy
Military
No research set
Whig Party
At peace
January 1, 1836
The United Kingdom
Death of Prin...
Brussels
Cambrai
Charleville
Verdun
Arlon
Frankfurt
Mainz
Cleves


OpenVic (DEBUG)
Production
Budget
Technology
Politics
Population
Trade
Diplomacy
Military
No research set
Whig Party
At peace
January 1, 1836
The United Kingdom
Death of Prin...
Aosta
Novara
Milan
Bergamo
Brescia
Trent
Verona
Treviso
Padua
Turin
Alessandria
Parma
Genoa
Massa
Bologna
Florence
Nice
Savona
Ligurian Sea

As you can see, the resource icons are now displayed on top of the provinces and additionally the nation capitals are also displayed. In game development, these are known as billboards: textures which use a shader (The shader tells the graphics card how to display your object in 3D) to always face the game's camera/screen. We use Godot's multimesh, which allows us to easily (and cheaply performance-wise) render hundreds to thousands instances of a single model in the game world, all with just a single draw call to the graphics card.

Upon game start, we find the city positions within each of the provinces and place a simple flat square model at those positions. We give a number to our billboards shader for each province, which tells it which picture among capitals, resources, crimes, etc. to display. This shader is also what tells the squares to face the camera and places them in front of everything else in the 3D world.


Administration: Sea power & The Merchant... Current research: No research selected

Army 5/30 | Navy 8/30 | Commerce 9/30 | Culture 5/30 | Industry 8/30

Naval Doctrine	Ship Construction	Naval Engineering	Naval Science	Naval Leadership
Post-Nelsonian Thought	Clipper Design	Naval Design Bureaus	Alphabetic Flag Signaling	The Command Principle
Battleship Column Doctrine	Steamers	Fire Control Systems	Naval Plans	Naval Professionalism
Raider Group Doctrine	Iron Steamers	Weapon Platforms	Naval Statistics	Naval Decision Making
Blue and Brown Water...	Steel Steamers	Main Armament	Naval Logistics	Naval Risk Management
High Sea Battle Fleet	Steam Turbine Ships	Advanced Naval Design	Naval Directionism	Naval NCO Training
Modern Naval Doctrine	Oil-Driven Ships	Modern Naval Design	Naval Integration	Modern Naval Training

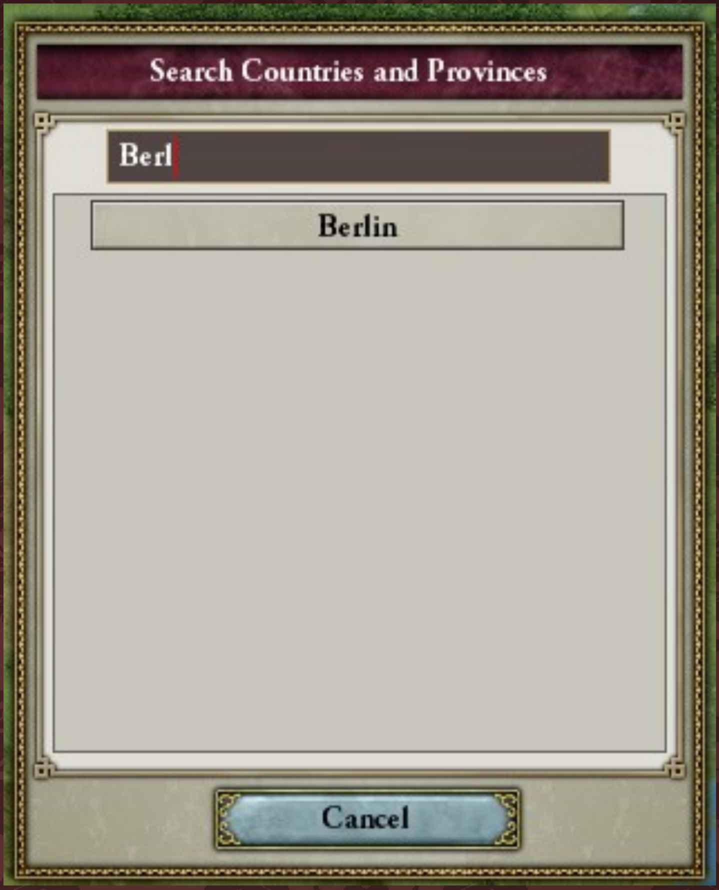
Possible Inventions | Name | %

None Selected | Research points: | Activation year:

Start Research

We've also been recently working on the UI for the Technology menu. While it still lacks some features such as image loading, descriptions of selected techs and displaying available inventions, we have successfully implemented tech schools, tech folders and tech completion status, all of which load and display in a dynamic fashion. This means that mods that add rows of techs or even whole new tech folders (such as Army, Navy or Industry as shown prior) will work out of the box in OpenVic.

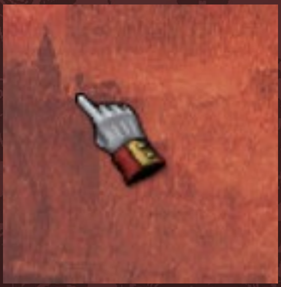
To further complement the Budget and Population topbar panel, we've fully implemented the search panel, allowing you to find and move to countries and provinces.

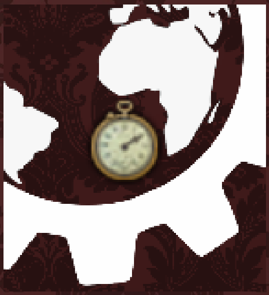


An additional upcoming UI update is the ledger, which will be a great way to review the accuracy of simulation data and visualise our progress.

Cursors:

We've also made a major UI update with the implementation of cursor loading from Vic2. There are two varieties of cursor, standard cursors and animated cursors (stored in the windows .cur and .ani format respectively).





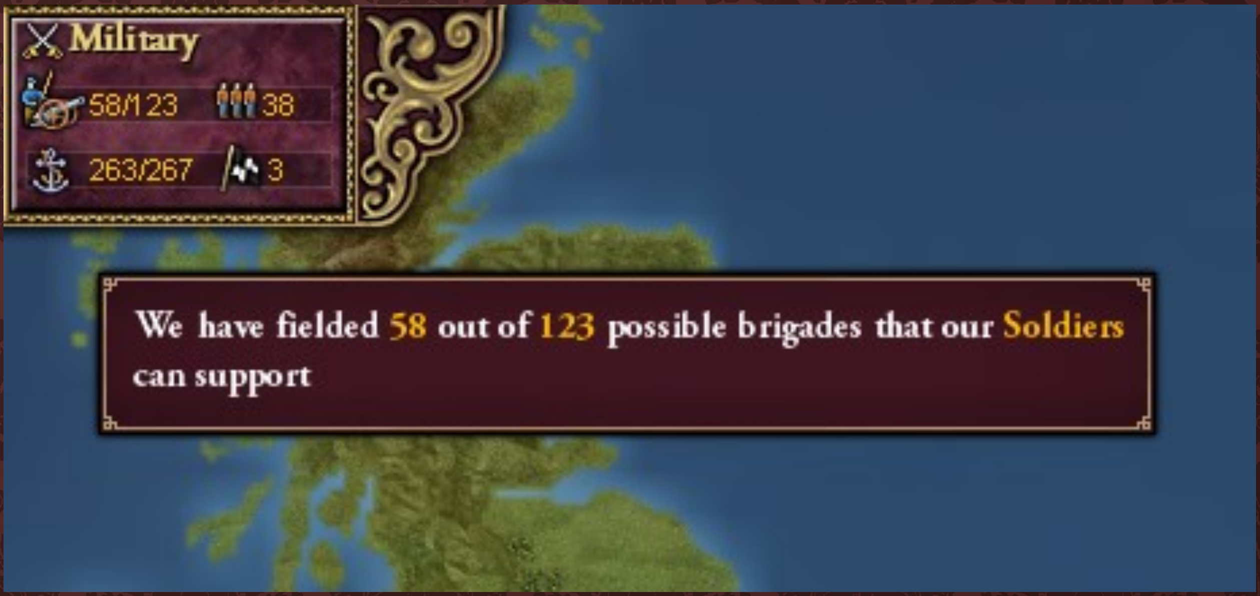
Godot doesn't support these image formats out of the box, so we wrote some code to load and store these images ourselves. Godot also lacks support for animated cursor textures, which necessitated us adding our own system to animate the cursors too.

In all, OpenVic now has working cursors integrated into the game, with the only remaining work being to tell the game when it should switch cursors as we implement more features such as unit selection.

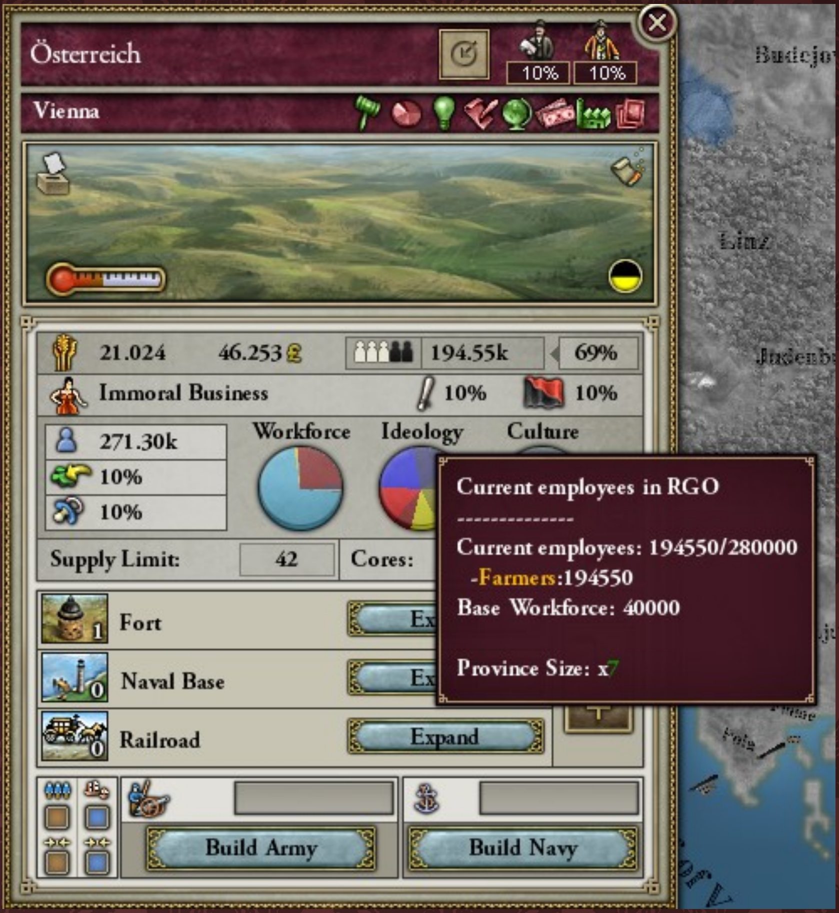
Tooltips and Labels:

Finally, we have fully integrated tooltips into OpenVic's UI. This includes all our custom UI element nodes which have helper functions, allowing easy updating of tooltip text which is automatically displayed as the mouse hovers over each node.

The tooltip itself uses information from the GUI defines files so that the texturing matches that of either vanilla Vic2 or a custom modded texture. We still need to write code to generate the specific text for each element's tooltip and update it as the game progresses, which we have already done for several menus, and we to plan to continue doing throughout UI development.



We have fielded 58 out of 123 possible brigades that our Soldiers can support



10%
10%
21.024
46.253
194.55k
69%
Immoral Business
10%
10%
271.30k
10%
10%
Supply Limit: 42
Cores:

Current employees in RGO

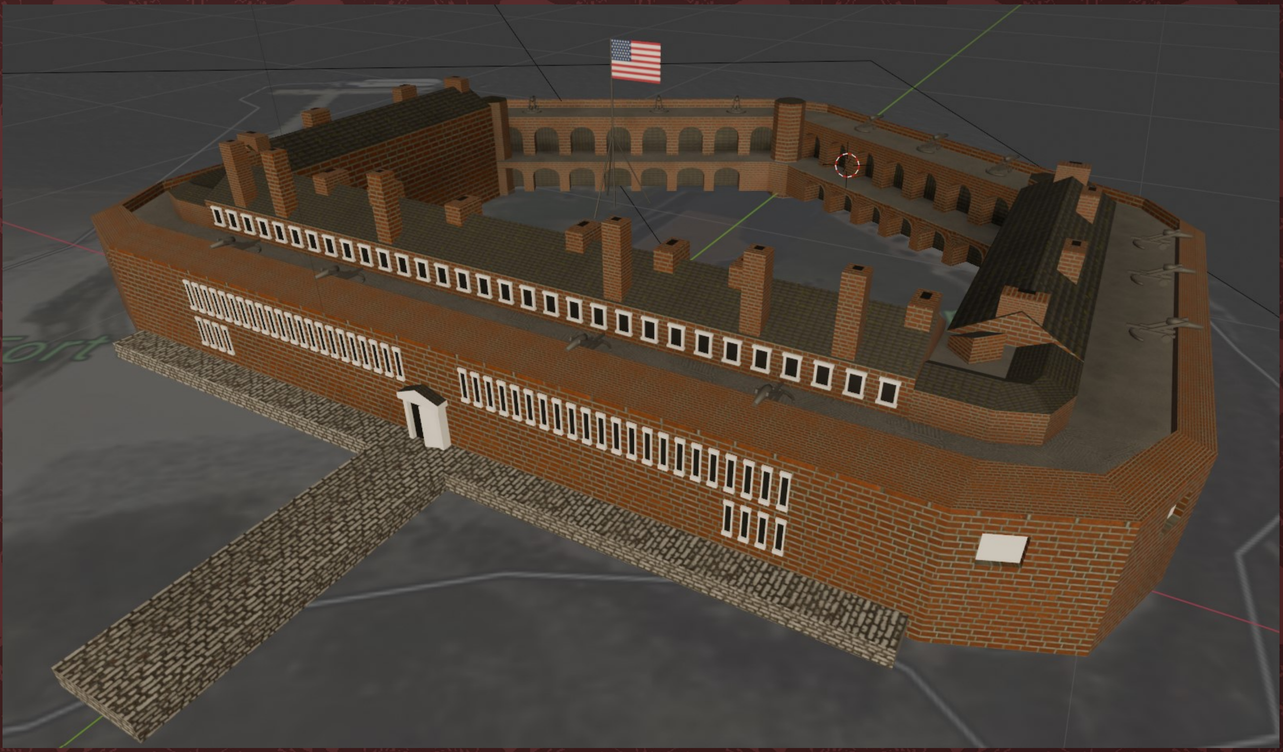
Current employees: 194550/280000
-Farmers:194550
Base Workforce: 40000
Province Size: x7

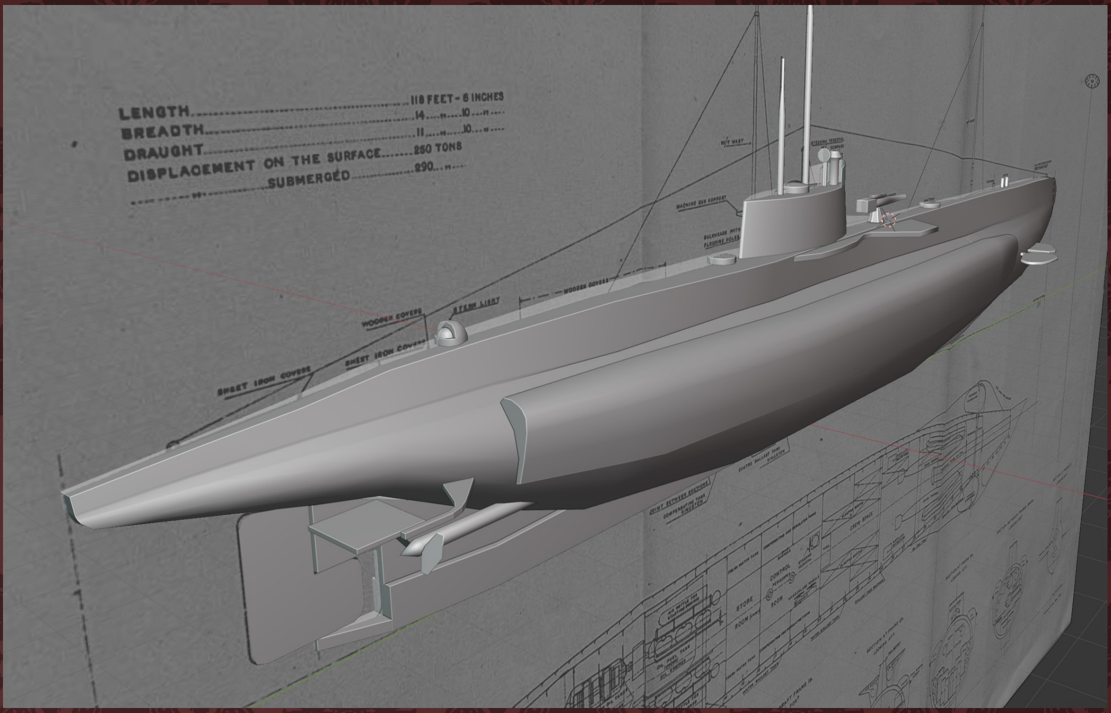
We have created a new custom label node for tooltips, which is also used for regular text labels throughout the UI, supporting Vic2's various localisation features, such as color codes (which are dynamically loaded from the defines files, and therefore easily modifiable), the special currency icon, and dollar-enclosed substitution values (such as "Current employees: $EMPLOYEES$/$EMPLOYEE_MAX$").

Art Team:

While much of our recent effort on OpenVic has been focused on emulating the intricate featureset of Vic2, our Art Team has slowly but steadily been churning away some custom assets for later integration into the game.

Take a look at some of the 3D models recently cooked up by Rataz:




LENGTH .... 118 FEET - 6 INCHES
BREADTH .... 14 .. " .. 10 .. "
DRAUGHT .... 11 .. " .. 10 .. "
DISPLACEMENT ON THE SURFACE .... 250 TONS
SUBMERGED .... 290 .. "
WOODEN COVERS
STERN LIGHT
SHEET IRON COVERS

Alongside this we've been planning UI graphics for entirely new menus we hope to add in the future as QOL features. Things should be interesting on this front, so stay tuned!

Conclusion:

And finally, we'd like to thank everyone following this project for being patient (our dev diary releases are gradually getting more and more spaced out we're realising).

Have no fear however, we've just heavily focused on building the internal mechanisms of the game, which while full of interesting and technical solutions, are often rather difficult to show off in their fullest in the dev diary format.

In fact, as we've been writing this Dev Diary, we've been asking ourselves whether the folks here are interested in increasingly technical sections on how we overcome these problems or if you would prefer if we simplied some of these sections to make future dev diary reading experiences more pleasant.

We're looking forward to sharing more with you as we continue to explore and understand the depths of Vic2's game systems and bring them to OpenVic!

As ever, thanks again for following our motley project and have a Happy new year!

The OpenVic Team