OpenVic

Dev Diary 7

Salutations!

Hello everyone, welcome back for the 7th OpenVic Dev Diary! This one is packed with pretty screenshots of models, maps and menus, as well as some discussion of the backend work that makes it all possible.

Let's begin with a quick overview of where we are in the project's development path. The parts needed to make OpenVic can be broken down into three broad sections:

- Dataloading - parsing, validating and reading into memory all the text and image files which define the game's content. This includes loading vanilla defines and any existing mod to the degree Victoria 2 supports it, with all the fun edge cases and error handling that involves.
- User Interface - building out the visual side of the game: the map with all its provinces, mapmodes and models, and the menus, every icon, textbox, tooltip, button and slider, all behaving as is familiar to us from the original game. Despite relying on the other sections for information to display, this is still a massive task due to the number of menus, their complexity and the need to ensure every component looks and functions like their Victoria 2 counterpart.
- Simulation - the core of the game, storing the game state and modifying it according to rules and mechanics. This includes POPs, the economy, AI, units, battles, events, decisions, and everything else that makes up the gameplay. It's both the most important and least well understood part of Victoria 2, and so the part of OpenVic that is most crucial to get right.

We finished the Dataloading section at the beginning of the year and since then have shifted focus on to the User Interface. We decided to prioritise this over the Simulation as it will be a lot easier to implement OpenVic's complex mechanics when we're able to see the results in game as we develop them. This also gives us more visible progress to share, rather than having everything in a user-unfriendly terminal until the very end.

We are currently midway through the User Interface section with the backend code for loading models and generating menus from text defines finished, leaving us to implement the scripts that use that code to generate and control the behaviour of specific menus as well as some further mostly cosmetic map and model work.

With that said, let's move on to the star feature of this Dev Diary and read about how Nemrav wrote our model loader.

Models

Victoria 2, like other Paradox titles from the same period, uses XAC and XSM files for 3D models and animations respectively. These were formats created as part of the EmotionFX suite by the now defunct Mystic Game Development. There is no official documentation online for them, but we were still able to find unofficial documentation dating to 2014 on Internet Archive's wayback machine. This documentation allowed us to get an initial prototype for a model and animation loader working in Godot as a separate side project.

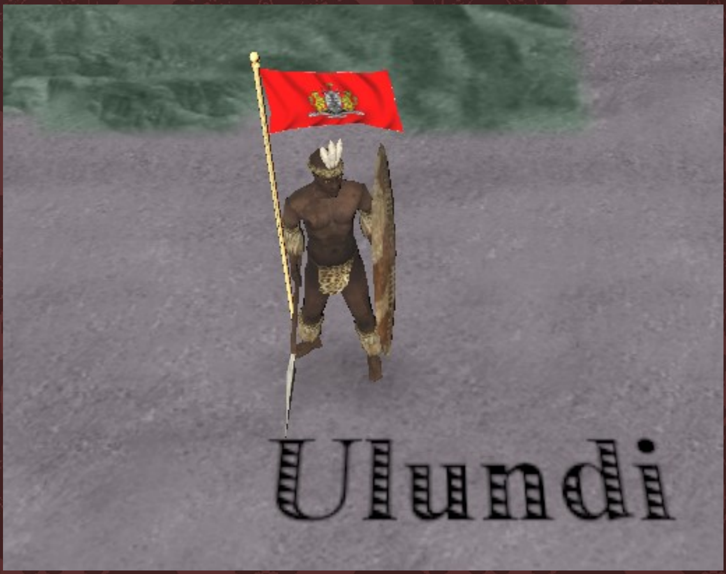
In essence, the file format is divided into chunks each with a type, length, and version. Depending on the chunk's type, it could contain metadata, vertex data, skeleton, or material (texture) information. Writing a loader was a matter of looking at the variables specified for each chunk in the documentation, then writing a function which would load that data into Godot and assign it to our own matching variables. Thankfully most 3D file formats are simply different ways of storing what is fundamentally the same kind of data (3D positions and rotations), so mapping this data to what Godot wanted was fairly simple.

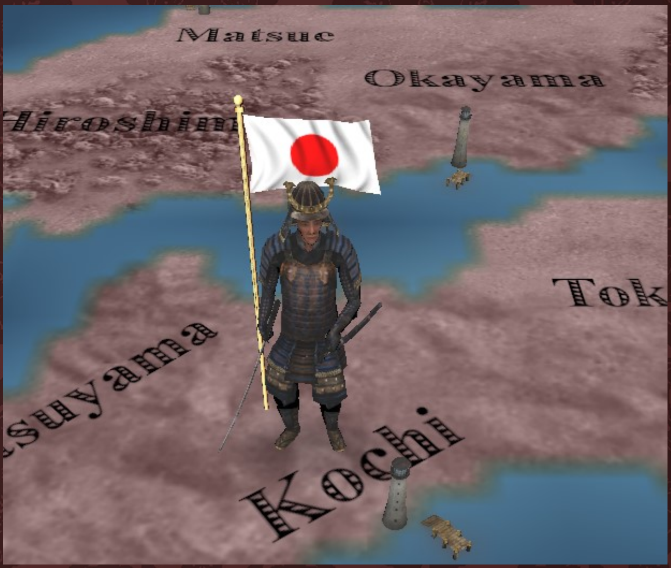
However, when testing that all models could be loaded, we noticed that the documentation for the model files we were using had to be incomplete because some models refused to load. These were due to some particular files using older chunk versions, and chunk types that the documentation didn't cover. After some investigation, we found that the older chunk versions simply lacked a variable or two (which weren't used anyways) and so were simple to fix. We have decided to skip the undocumented chunk types for the time being, since to our knowledge they are not needed to display the models correctly. Paradox didn't use all the features that EmotionFX provided them, so this is hardly surprising. Nevertheless, if we do find proper documentation for these and their functionality at a later date and we find they do make a difference, we will return to this and see about implementing them.

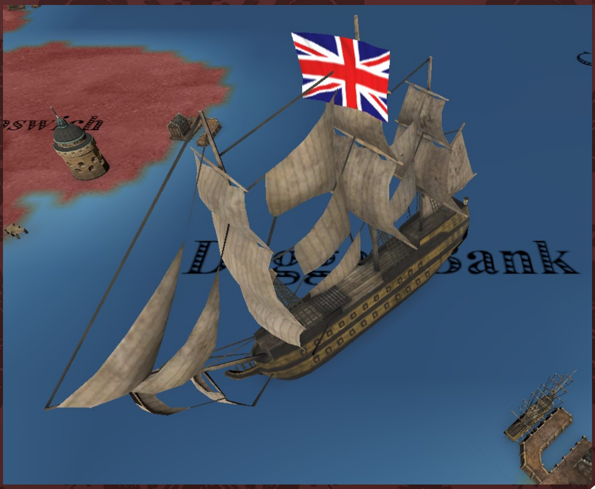
Rotterdam
Utrecht
Bank
Middelburg
Eindhoven
Hasselt
Maastricht
Cleves
Brussels
Spa
Aachen
Siegburg
Namur

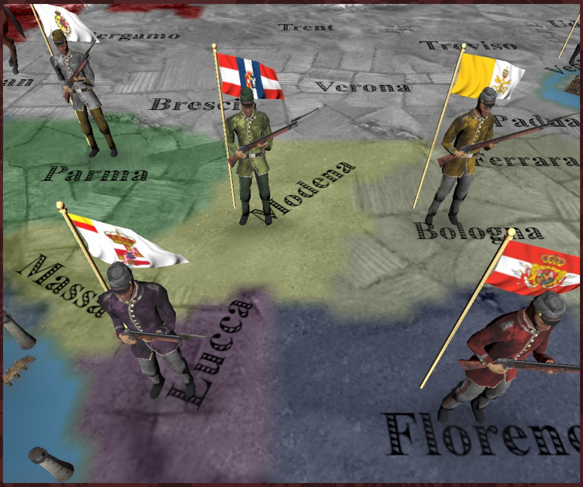
Satisfied with our prototype, we then integrated the model and animation loader into the main project and used it to load "actors" defined in GFX text files. These include buildings like forts and ports/naval bases, with variants for different building levels, as well as a wide variety of military unit models, both land and naval. We support loading a specific model and idle, move and attack animations for each unit and graphical culture type combination, with infantry (for land units) and the generic graphical culture type as fallbacks when specialised models are missing.

We also load and attach guns and helmets chosen based on graphical culture type, as well as extra attachments specified in the actor's GFX file definition, for example Zulu infantry have a shield, as well as their cultural "gun" variant (a spear). Mounted units are a special case in which the horse is generated as the base model with the soldier as an attachment, with alternate animations to make the soldier ride, rather than stand, on the horse. Each unit comes with a flag, either on a pole beside it or floating above it, rendered using a large texture made by stitching together each flag variant for every country in the game.



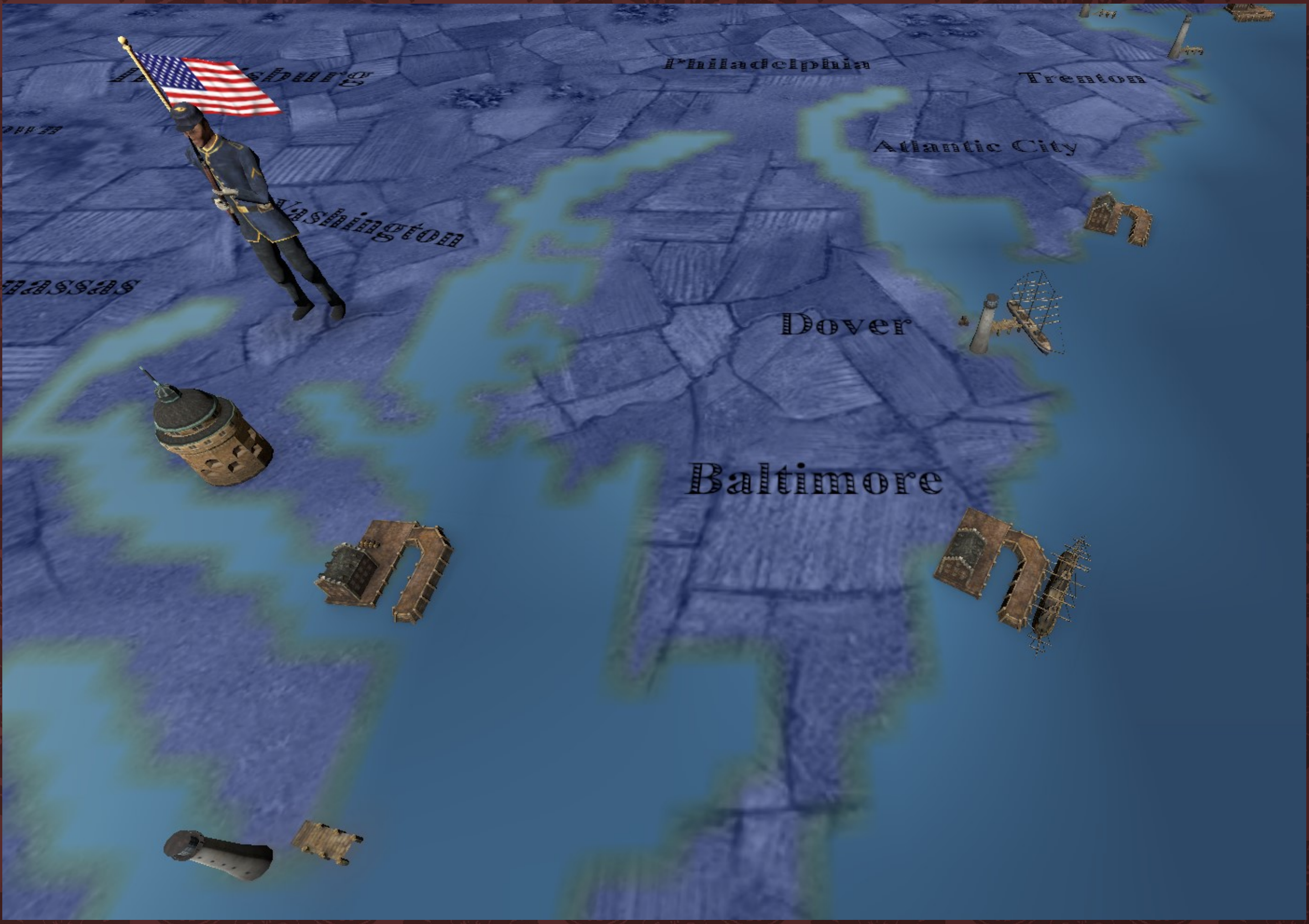




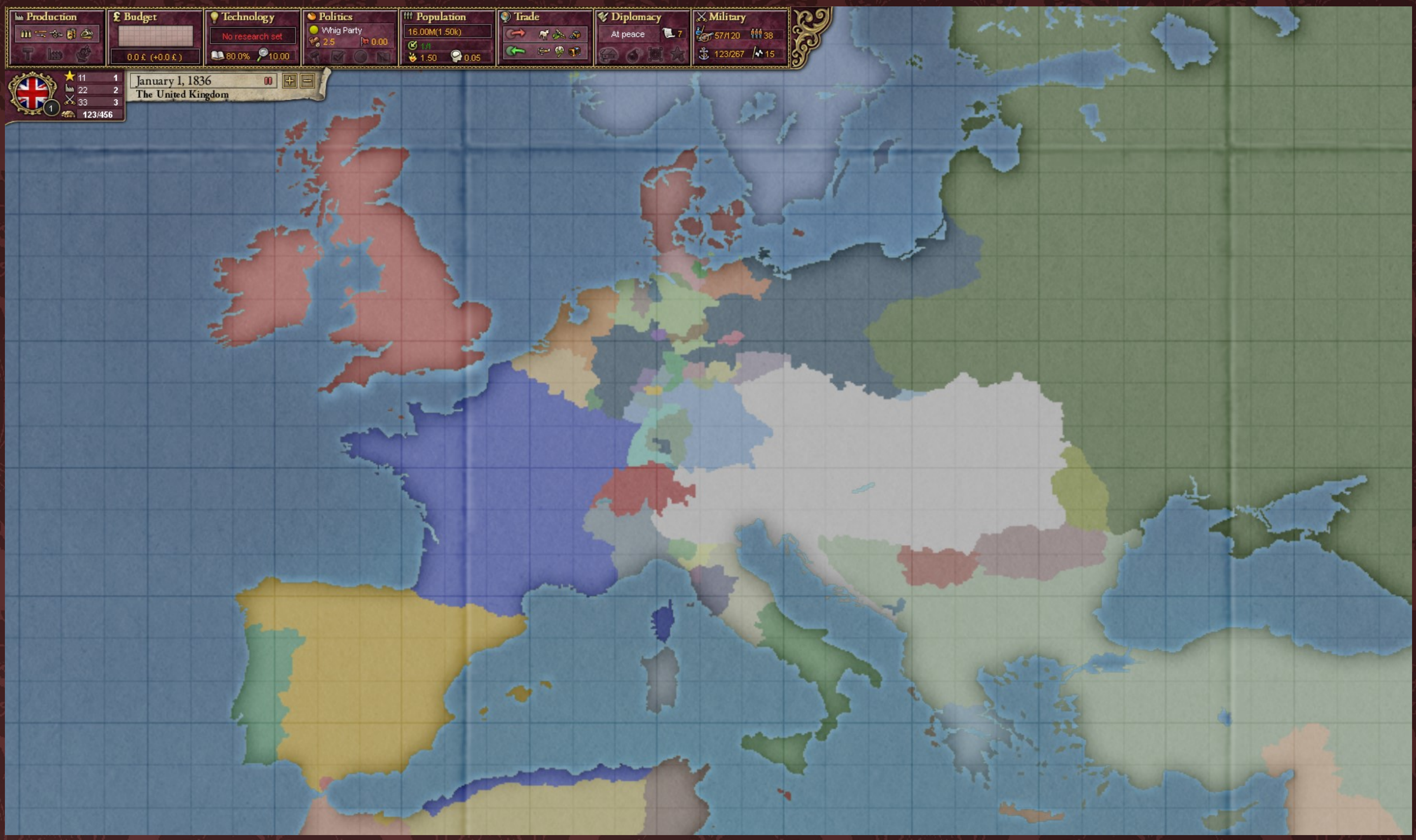


Map

Our map has received a lot of care since the last dev diary. Province colours now match their Victoria 2 equivalents, putting an end to the off-putting slightly-too-bright colours of the map's earlier iterations. The terrain textures have been improved, with better placement, scaling, blending and visibility in coloured provinces, as well as geographical tinting to give some variety despite using the same terrain types across the map. Water is no longer a single solid shade of blue, but is instead textured with bright coastlines and varying tint across the map. And of course, provinces now have names, which does a lot to make OpenVic look and feel like Victoria 2.



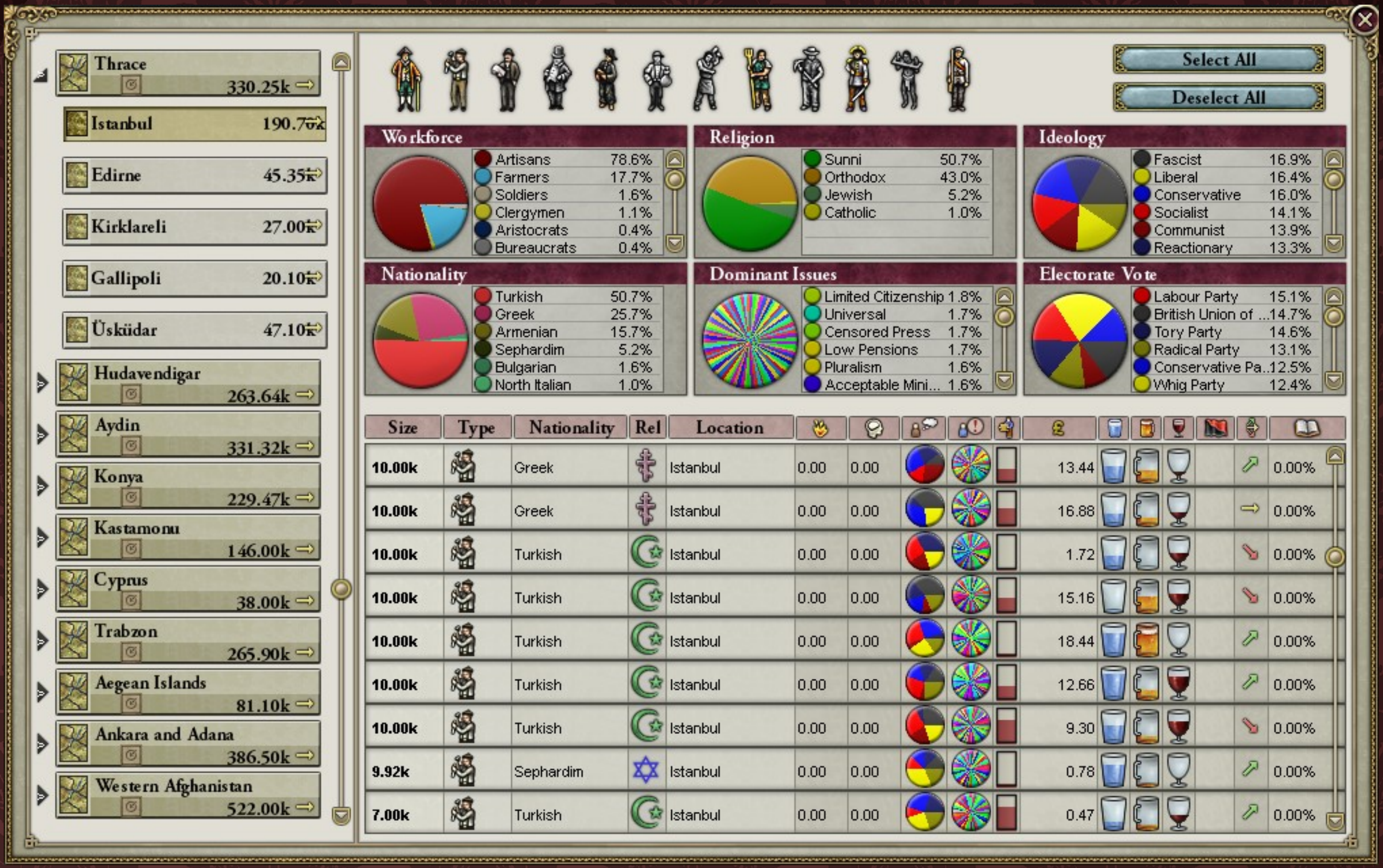
As you zoom out, the terrain of the detailed map is replaced with a flat, parchment like texture, with shaded coastlines adding to the antique map style (and hopefully compensating for the borders I need to get around to adding).



Menus

In the last Dev Diary we saw some early examples of menus and discussed the system for generating them from GUI and GFX files. Since then the menu generating system has been developed further to support more elements including sliders, scrollable lists, and overlapping icons (e.g. for province cores and modifiers). More menus have been added, in particular the nation management screens, with accompanying scripts to react to input and update their contents.

The most complex of these is the Population Menu, with around 1.5k lines of supporting C++ and GDScript code. It has all the main features of Victoria 2's version, with an expandable and selectable state and province list, pop type filter buttons, and pie charts showing the distribution of pop types, cultures, religions, issues, ideologies and votes of the selected pops. The pop table is packed with information including size, culture and location labels, pop type and religion icons, vote and issue pie charts, and unemployment and needs progress bars.



Province	Population
Thrace	330.25k →
Istanbul	190.70k
Edirne	45.35k →
Kirklareli	27.00k →
Gallipoli	20.10k →
Üsküdar	47.10k →
Hudavendigar	263.64k →
Aydin	331.32k →
Konya	229.47k →
Kastamonu	146.00k →
Cyprus	38.00k →
Trabzon	265.90k →
Aegean Islands	81.10k →
Ankara and Adana	386.50k →
Western Afghanistan	522.00k →

Workforce	
Artisans	78.6%
Farmers	17.7%
Soldiers	1.6%
Clergymen	1.1%
Aristocrats	0.4%
Bureaucrats	0.4%

Religion	
Sunni	50.7%
Orthodox	43.0%
Jewish	5.2%
Catholic	1.0%

Ideology	
Fascist	16.9%
Liberal	16.4%
Conservative	16.0%
Socialist	14.1%
Communist	13.9%
Reactionary	13.3%

Nationality	
Turkish	50.7%
Greek	25.7%
Armenian	15.7%
Sephardim	5.2%
Bulgarian	1.6%
North Italian	1.0%

Dominant Issues	
Limited Citizenship	1.8%
Universal	1.7%
Censored Press	1.7%
Low Pensions	1.7%
Pluralism	1.6%
Acceptable Mini...	1.6%

Electorate Vote	
Labour Party	15.1%
British Union of ...	14.7%
Tory Party	14.6%
Radical Party	13.1%
Conservative Pa..	12.5%
Whig Party	12.4%

Size	Type	Nationality	Rel	Location						£						
10.00k		Greek	‡	Istanbul	0.00	0.00				13.44					↗	0.00%
10.00k		Greek	‡	Istanbul	0.00	0.00				16.88					→	0.00%
10.00k		Turkish	☪	Istanbul	0.00	0.00				1.72					↘	0.00%
10.00k		Turkish	☪	Istanbul	0.00	0.00				15.16					↘	0.00%
10.00k		Turkish	☪	Istanbul	0.00	0.00				18.44					↗	0.00%
10.00k		Turkish	☪	Istanbul	0.00	0.00				12.66					↗	0.00%
10.00k		Turkish	☪	Istanbul	0.00	0.00				9.30					↘	0.00%
9.92k		Sephardim	✡	Istanbul	0.00	0.00				0.78					↗	0.00%
7.00k		Turkish	☪	Istanbul	0.00	0.00				0.47					↗	0.00%

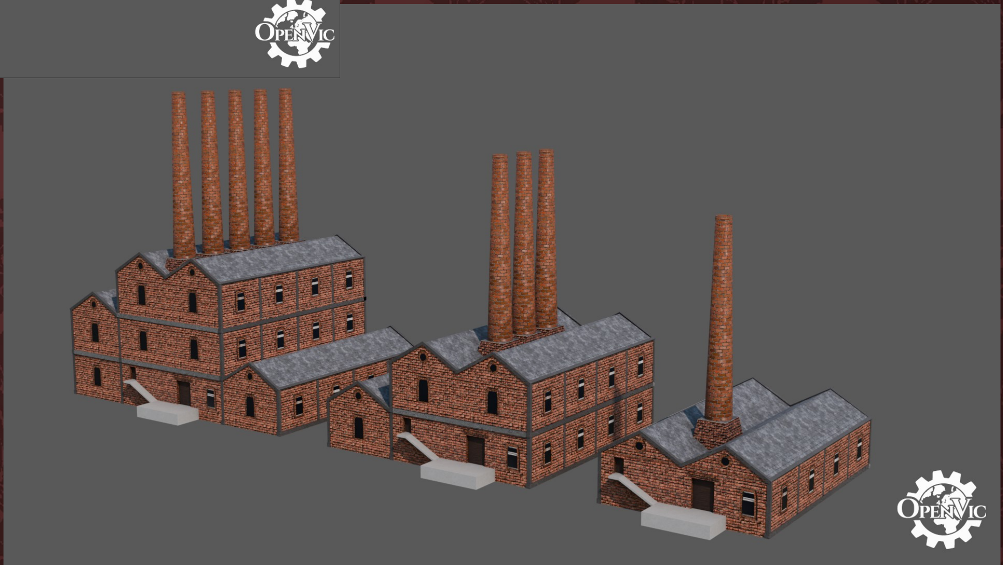
Although some of this information is based on placeholder data, for example each pop’s issues, ideologies and votes are currently randomised, everything beyond that is working in its finished state carrying the data up from the C++ simulation to the Godot menu. All of this is supported by an intricate internal system that collects and filters selected pops, calculates statistics about them, and makes all of the data available in a performant manner - rather than sending all the information for the currently selected pops up from the C++ layer to the GDScript layer, we carefully keep track of what’s currently visible and send only what’s needed. This means the menu can easily handle having all the world’s pops selected at the same time, as they are by default in our current dev build.

Art Team

With a solid framework of the game now firmly before us, the Art Team has begun drafting concepts for original OpenVic custom assets. The recent progress of the Art Team can be divided into two categories, the UI concept art and the 3D models.

We've recently scoped out our UI requirements, and have begun developing them from there, while our 3D modelling scope is still being drafted. We've explored a number of UI concepts, looking at ways to improve both the visual presentation and function enhancements to the user experience, such as the Additional Info Bar concept. There have been discussions on a Multiplayer Server browser and the forms that it would take, alongside plans for a Mod Selection Menu.

Our 3D Modellers have also been hard at work, and have produced a substantial number of custom assets for us recently. Thanks to this deluge of custom additions, (such as a wider variety of unit models or factory levels visible on the map) that could be added to OpenVic.



Victoria 2 Directory Selection

If you want to try out OpenVic for yourself you can download our latest release from Github and run the executable for your operating system. If you move the executable, make sure to move the other files in the folder as the program needs them to run. You'll need to have Victoria 2 installed, as OpenVic reads its define to provide the same content and graphics as vanilla Victoria 2. OpenVic will try to find the install folder as described in the steps below, bit if it can’t be found automatically you’ll be asked to select the folder through a menu when first running the game.

This only needs to be done once, as after that the path will be saved for future use.

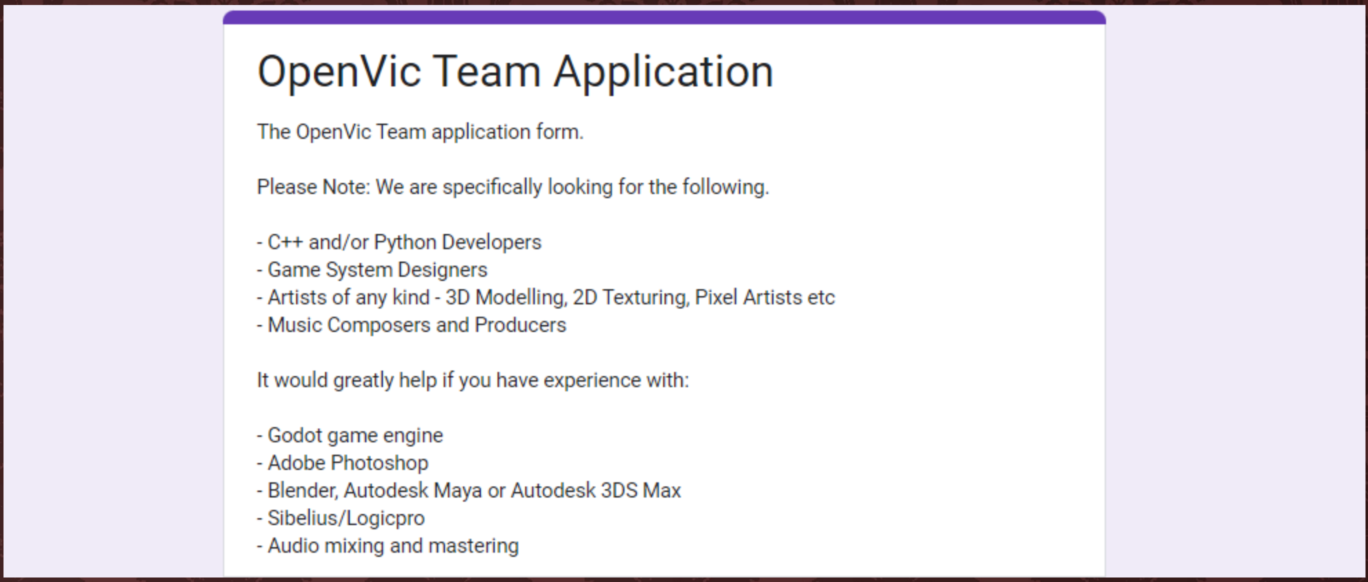
How OpenVic finds your Victoria 2 Install:

1. If you supplied this path (using a command line argument `-s “<path>“`) to a folder named “Victoria 2” that contains “v2game.exe” in it. It then uses that path (OpenVic doesn't use the exe, we just check that it exists to confirm we’ve found the correct directory), else
2. Checks if the directory the OpenVic executable is placed in is a valid Victoria 2 install following the rules of step 1 , else
3. Searches for your Steam programs directory and checks for a Victoria 2 install following the rules of step 1, else
4. It will ask the user to supply the path to Victoria 2 manually.

Once the path is found, it is stored in the game settings so that these checks would not have to be performed again.

Conclusion

As we are coming to an end with this diary, we wanted to thank everyone for their support and patiently waiting for this milestone conclusion of the project. If we haven't bored you out with these technical details and you also share our enthusiasm for constructing this monumental work, then please feel free to submit your application here.



OpenVic Team Application

The OpenVic Team application form.

Please Note: We are specifically looking for the following.

- C++ and/or Python Developers
- Game System Designers
- Artists of any kind - 3D Modelling, 2D Texturing, Pixel Artists etc
- Music Composers and Producers

It would greatly help if you have experience with:

- Godot game engine
- Adobe Photoshop
- Blender, Autodesk Maya or Autodesk 3DS Max
- Sibelius/Logicpro
- Audio mixing and mastering

We would love to hear from you and see how your talents can help the team achieve greatness!

OpenVic

As ever, thanks again for following our motley project and see you in the next one!

The OpenVic Team