Friday Facts #277 - GUI progress update

kovarex, Klonan

2019-01-11

GUI progress update (kovarex)

This is a continuation of the last status report from FFF-269 . As it might not be a surprise, the biggest bottleneck of the 0.17 release is the GUI. I like to believe, that we have learned a lot from the pitfalls of the collaborative creative process of GUI. This is the typical way we were redesigning the GUI :

— Two to three people started discussing what could be cool to change in the particular GUI. Some people randomly joined and left the ongoing discussion. Arguments to discard certain ideas have to be repeated over and over. Then the discussion is ended because of something.
— A week later people start talking again, most of them forgot most of the stuff, or were discussing it with different people, so they assume some details of the changes to be understood by everyone, while they aren't.
— They come to an agreement how it should be done.
— They have a random discussion about it a week later and figure out, they had completely different ideas about how it should be done, they just didn't articulate them precisely. Both are kind of angry to have to reopen and renegotiate the subject again.
— Someone starts to implement the GUI, but half-way through it is uncovered, that there was another layer of misunderstanding when specifying how should the work be done, and we need to go to step 1 again and repeat.

Since many GUIs are thought and worked on in parallel, these situations overlapped and amplified the problems of mixing things up in our heads about what we agreed on in which GUI.

Luckily, we eventually figured out, that it can't be done like this, and since there is a lot of work in the GUI, we need to make a process. It goes like this :

— First, there is some general discussion about the GUI, all team members can share their ideas.
— kovarex + Twinsen sit alone in the office, and discuss for some time (can be hours), all the pros and cons of how things should be done, and make some agreement.
— Twinsen writes a detailed UX document about the GUI containing the structure, and more importantly the behaviour, in a detailed manner.
— Twinsen + kovarex discuss the UX document and propose changes until they agree on the final version.
— Albert + Aleš take the UX document and create a UI mockup based on it.
— kovarex + Twinsen + Albert agree on the UI mockup or propose changes.
— Someone is assigned to implement the GUI based on the UX document and UI mockup
— kovarex reviews that the implementation is correct and points out some inconsistencies that he can see. Part of this step is making sure, that we share as many GUI styles and code as possible across different GUIs.
— kovarex + Albert have a final look on the implementation and fix final details until they both agree that the screen is fully finished.

Having the UX documents/UI mockups always available proved to be a huge time saver. Not only it helps us to solve the communication problems, we also don't have to remember and re-articulate decisions from some time ago as we can just open the document and see what we agreed on and instantly continue where we left off.

A good part of this strict pipeline is that we now have better knowledge of the state of the work progress.These are the GUI screens that we hope to deliver for 0.17 :

	General UX	UX draft	UX review	UI mockup	UI review	Implementation draft	Implementation review	Final review
Load map	✓	✓	✓	✓	✓	✓	✓	✓
Save map	✓	✓	✓	✓	✓	✓	✓	✓
Graphics settings	✓	✓	✓	✓	✓	✓	✓	✓
Control settings	✓	✓	✓	✓	✓	✓	✓	✓
Sound settings	✓	✓	✓	✓	✓	✓	✓	✓
Interface settings	✓	✓	✓	✓	✓	✓	✓	✓
Other settings	✓	✓	✓	✓	✓	✓	✓	✓
Map generator	✓	✓	✓	✓	✓	✓	✓	✓
Technology GUI	✓	✓	✓	✓	✓	✓	✗	✗
Technology tooltip	✓	✓	✓	✗	✗	✗	✗	✗
Recipe/item/entity tooltip	✓	✓	✓	✓	✗	✗	✗	✗
Action bar	✓	✓	✓	✓	✓	✓	✗	✗
Shortcut bar	✓	✓	✓	✓	✓	✗	✗	✗
Train GUI	✓	✓	✓	✗	✗	✗	✗	✗
Manage/Install mods	✓	✓	✓	✓	✓	✗	✗	✗
Main screen chat	✓	✓	✓	✗	✗	✗	✗	✗
Recipe explorer	✓	✓	✓	✗	✗	✗	✗	✗
Character screen	✓	✗	✗	✗	✗	✗	✗	✗
Menu structure	✓	✓	✓	✗	✗	✗	✗	✗
New game	✓	✓	✗	✗	✗	✗	✗	✗
Help overlay	✓	✓	✓	✗	✗	✗	✗	✗
Chat icon selector	✓	✗	✗	✗	✗	✗	✗	✗
Blueprint library	✗	✗	✗	✗	✗	✗	✗	✗

You can see, that there is still a lot of to do, but the work tends to accelerate as more and more of the GUI layouts/tilesets/standards are being finalized and reused. The conclusion is that 0.17 experimental in January is possible, but it might be February as well :).

The little details (kovarex)

Also, one of the reasons the work progresses slower, is that since we are making final versions of everything, we want to make it feel polished before we consider it finished, since there won't be time to come back to it.

For example, in every settings screen, we have the 'reset to default' button now. The first logical step was to only enable the button, when some of the options are different from the defaults. But the next logical step was to somehow let the user know, what settings will be changed when he uses the reset button. So we simply highlight all the non-default settings when hovering the reset button, and it feels nice since suddenly you have instant feedback of what you can expect from the action.

Loading Video

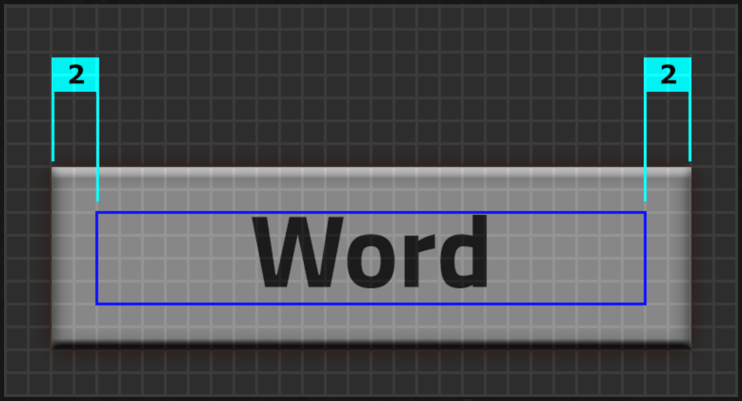
I understand, that spending too much time in settings GUI might not look like a good idea, since it is not used that much and the in-game screens are more important, but many of these principles we realize on the way will be useful when designing changes for the in-game GUI.

The scaling problem and solution (kovarex)

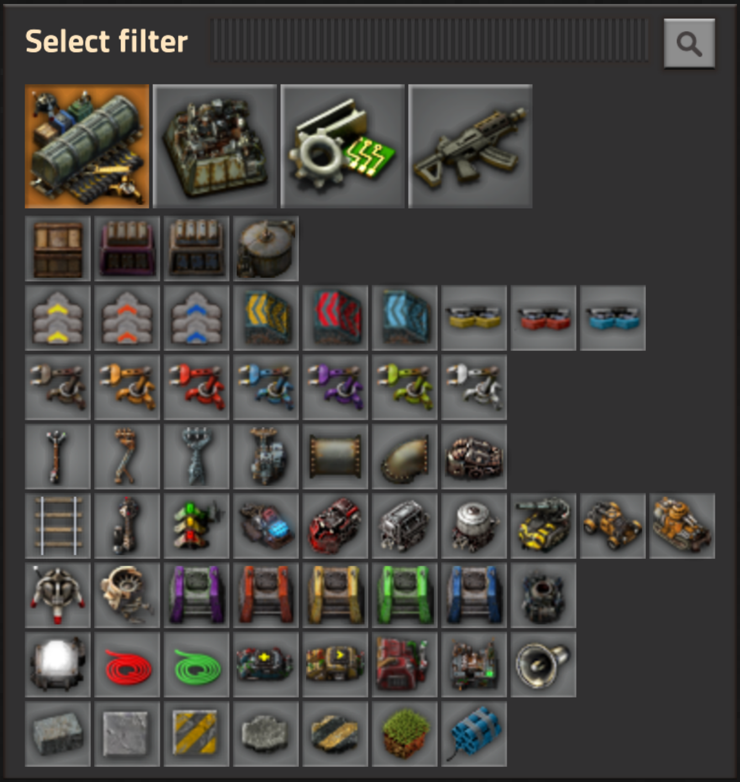
One of the many goals of the GUI rewrite was to make sure that the GUI looks correct in all possible UI scale values. By correct, we mainly mean, that the proportions of everything stay the same, so it is just smaller, but not deformed in a weird way. This might look like a simple thing to do, but it isn't as all the GUI values (sizes, widget positions, paddings etc.) are integer values, as in the end, every element needs to be on some specific pixel as long as we want it to be crispy clear.

Imagine that you have a 1 pixel wide gap between two 20 pixel wide buttons and then you want it to show in 120%. The buttons are enlarged to 24 pixels, but the gap either stays 1 pixel or becomes 2, but the proportions of the layout changes.

To solve this (and other issues), we decided to use something we call "modules". 1 module is 4 pixels in standard scale (100%), and almost everything is a multiplication of modules (sizes, positions, paddings etc). On top of that, we limited the possible scale values to be multiples of 25% (from 75% to 200%). This means, that the size of one module can be different (from 3 pixels at 75% to 8 pixels at 200%) but the proportions of everything stay the same, so it looks correct.



This works quite well so far, but it brings another problem for 0.17. I don't know if anyone noticed, but the item slots (inventory/logistic filters, crafting slots etc.) were intentionally scaled less than other UI elements. The reason for this was that the 32x32 icons we have for all the items/fluids/recipes don't look good if stretched too much.



32x32 icons that are stretched to 200% don't work good.

But since we had to abolish this special rule for 0.17 we need to make sure that all the 32x32 icons (285 items, 27 signals, fluids and more) will have to be provided in 64x64 resolution, so all the supported UI scales will look nice. This is going to be quite a lot of work, but since we render the icons from 3D models anyway, it should be manageable. We will probably push the high-res icons to 0.17 during the experimental phase.

Procedural Wave defense (Klonan)

We have had the Wave defense scenario in the game for a few years now, and over that time I have collected a lot of feedback. One problem I have determined, is that the scenario really lacks replayability, due to the fixed map. Since the map doesn't change at all, once you have a set of blueprints and tactic that works, repeat plays are mostly boring.

With recent changes and the great work by TOGoS, the procedural map generation is working really well, and is very reliable for a given set of settings. This gave me the idea, that it might be possible to make the Wave defense use a new map every time. I have been experimenting with some preset values, and I believe it will work really well.

However I have some indecision within myself, and there are several advantages and disadvantages between a handmade custom maps and randomly generated worlds.

— The map can be specifically designed and tweaked for a better experience, I can test and iterate the way biters move through the map, adjust the placement of trees, resources, tune the difficulty etc.
— We don't have to worry about map generation engine changes breaking the scenario.
— It is a reliable experience for all players, people can share specific tips, designs and tactics that are more specific for the map.
— The scenario has much greater replayability.
— We don't have to worry about migrating map data, tiles, entities, etc. to newer versions.
— Any improvements to the map generation will be reflected in the scenario.
— People can't cheese the scenario using other peoples blueprints.
— We can add some configuration options for people who want to tailor the experience.
— It is easy to add support for servers to continue running after victory/defeat.

So I am making the changes now to test whether procedural can work,and it shouldn't take too long as most of the scenario script will remain the same. I wanted to ask for some community feedback and thoughts : Have many of you played the Wave defense ? Do you think a random map would be more fun ? Do you think you would play it more if the map was different each time ?

As always, you are welcome to let us know what your think on our forum .