PROJECT HIGHRISE™ by SomaSim

Project Highrise Modding Part 3 - Small Office

Hello there! If you've already completed part 1 and part 2, how about something more challenging? In this tutorial we're going to make a new small office.

Compared to decorations, offices are quite a bit more complicated, because they have multiple moving parts that have to work together:

- They have workers, and those workers have work schedules and their own settings
- They need to be hooked up to tenants wait lists and advertising
- Just like decos, they have a visual layout, but it also needs to display "grime"
- Unlike decos, they also have an "under construction" visual state.

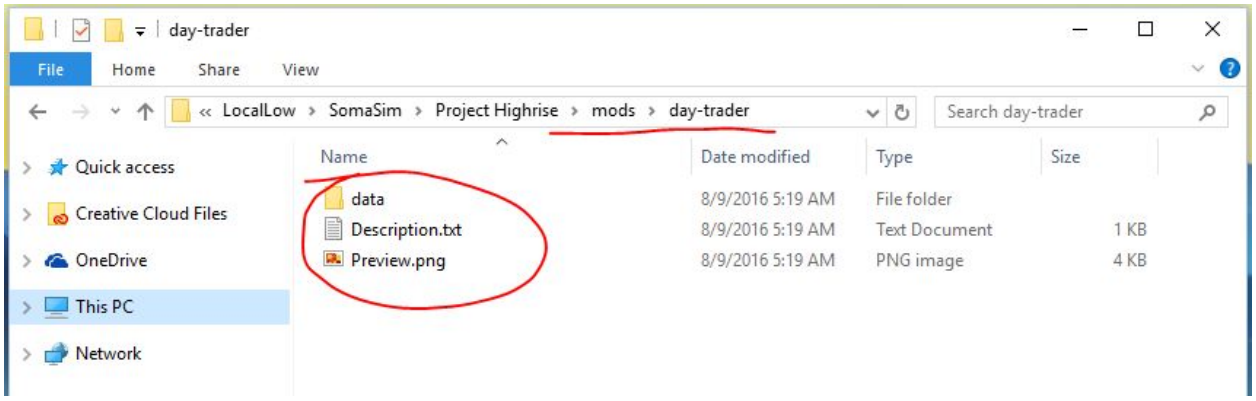
As before, we're going to start from a template, and then use that to discuss how it's put together as we change it.

Step 1. Getting a sample office

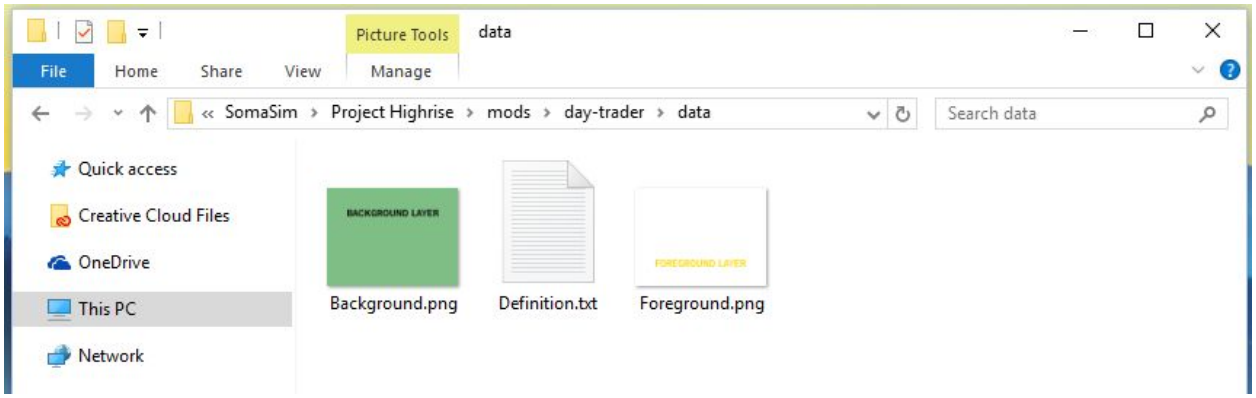
We're going to make an office mod for a one-person stock day trading company. These guys don't need much, just a good computer, so that fits perfectly, since they'll fit into a small windowless office. :)

As we've done before, make a new folder inside **mods**. I'm calling mine **day-trader.** Then download this small office template file and unzip it into your new folder: https://storage.googleapis.com/highrise-modding/templates/office-small-template.zip

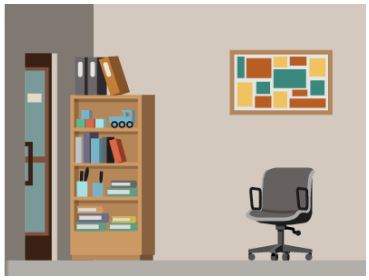
You should have a directory that looks like this:



Once again, go into your data folder, and see the sample images included in the template.



Let's replace them with something better. I used the free modding props from here to put together a little office scene:



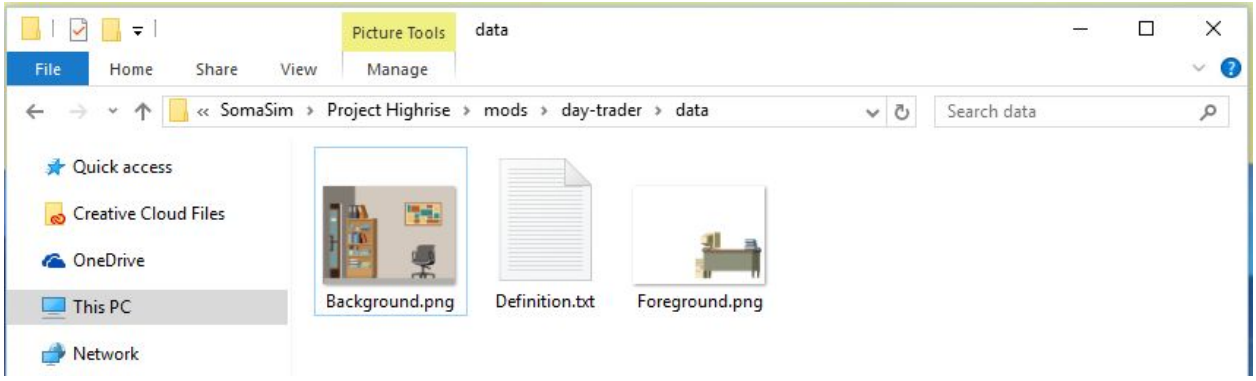
Background.png

Foreground.png

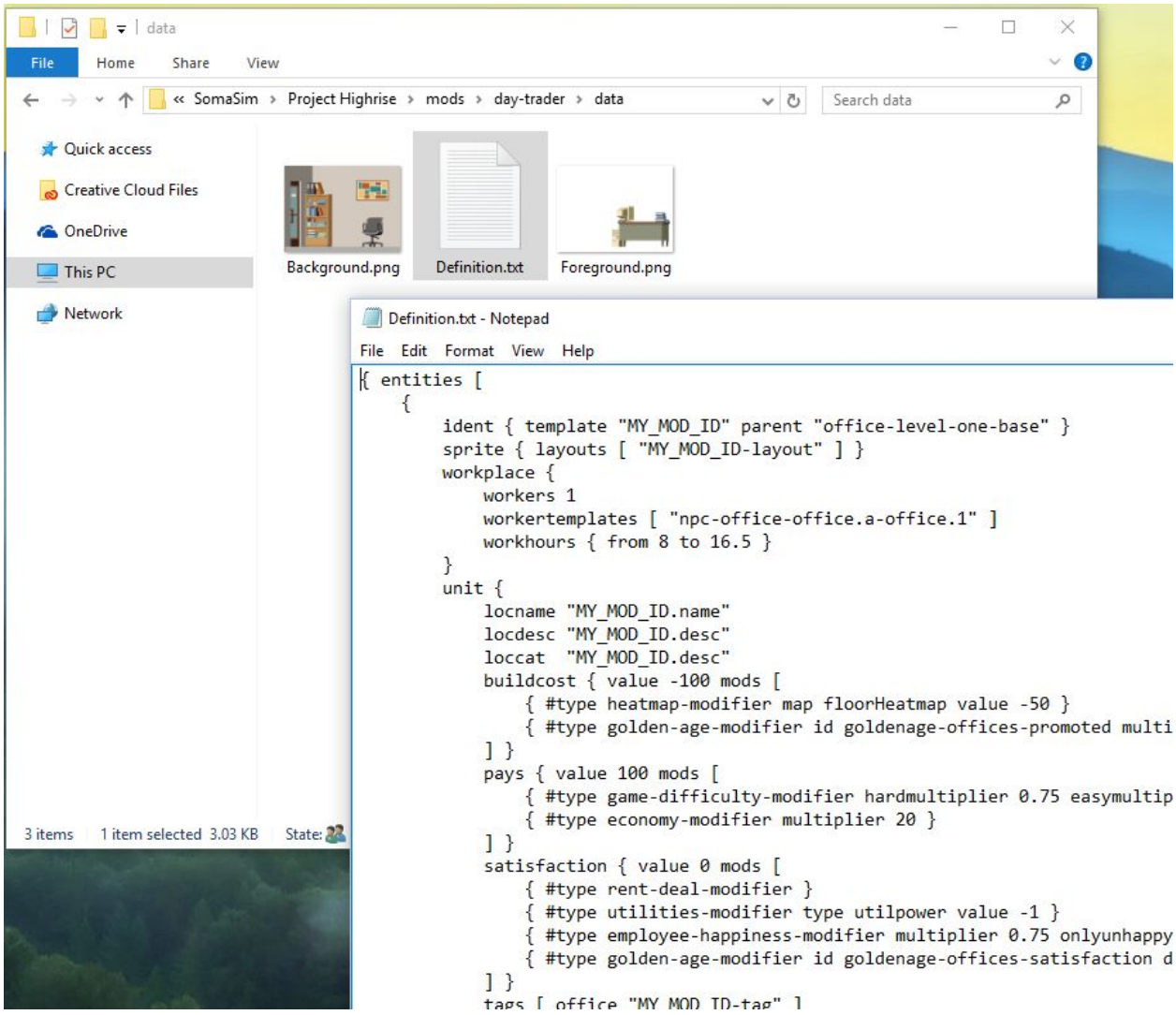
(Click on the links above to download them - or make your own different ones!)

Both of these images are 400px wide and 300px tall. A small office is always 4 tiles wide and 1 tile tall, and each "tile" is 100px wide and 300px tall, so that all adds up.

Now our folder should look like this:

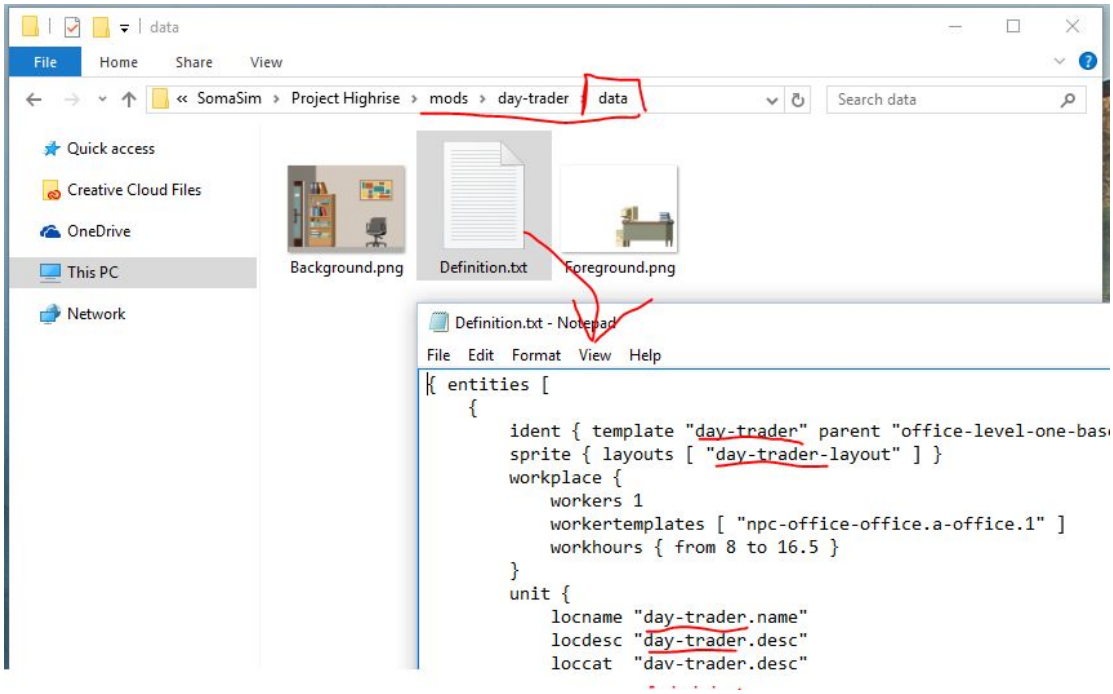


So far so good! Now open **Definition.txt** - you'll see it's quite a bit larger than what we saw with decos:



Once again, we're going to start with the following:

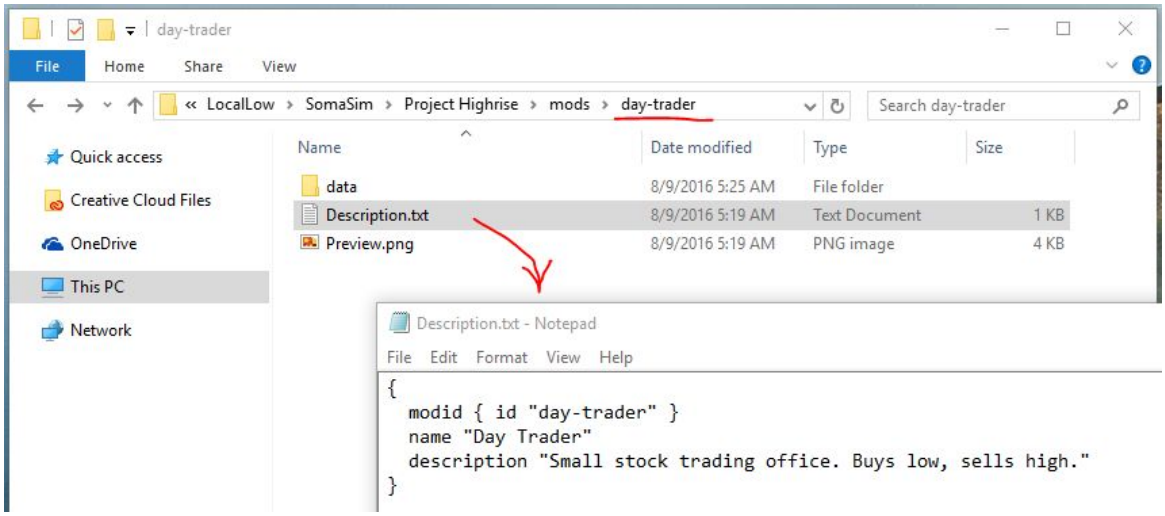
1. As before, in the **Definition.txt** file, search and replace **MY_MOD_ID** with our new id, which is **day-trader**



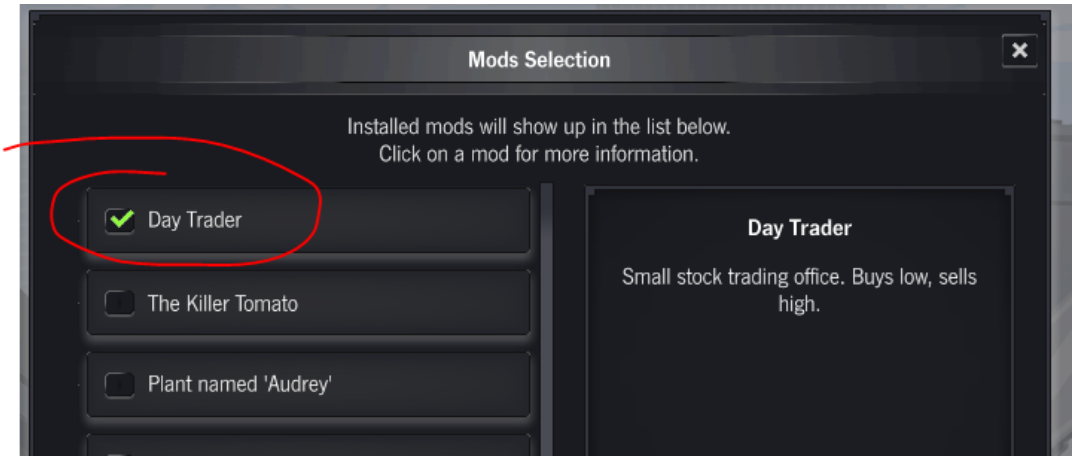
2. Again in the **Definition.txt** file, scroll down to the “text” section and replace the name and description as follows. Also, see the new “day-trader-movein.name” string? Replace that as well, so the entire section looks like this:

```
text {
    en {
        "day-trader.name" "ABC Day Traders"
        "day-trader.desc" "Stock trading office, buys low and sells high"

        "day-trader-movein.name"   "Day Traders"
    }
}
```

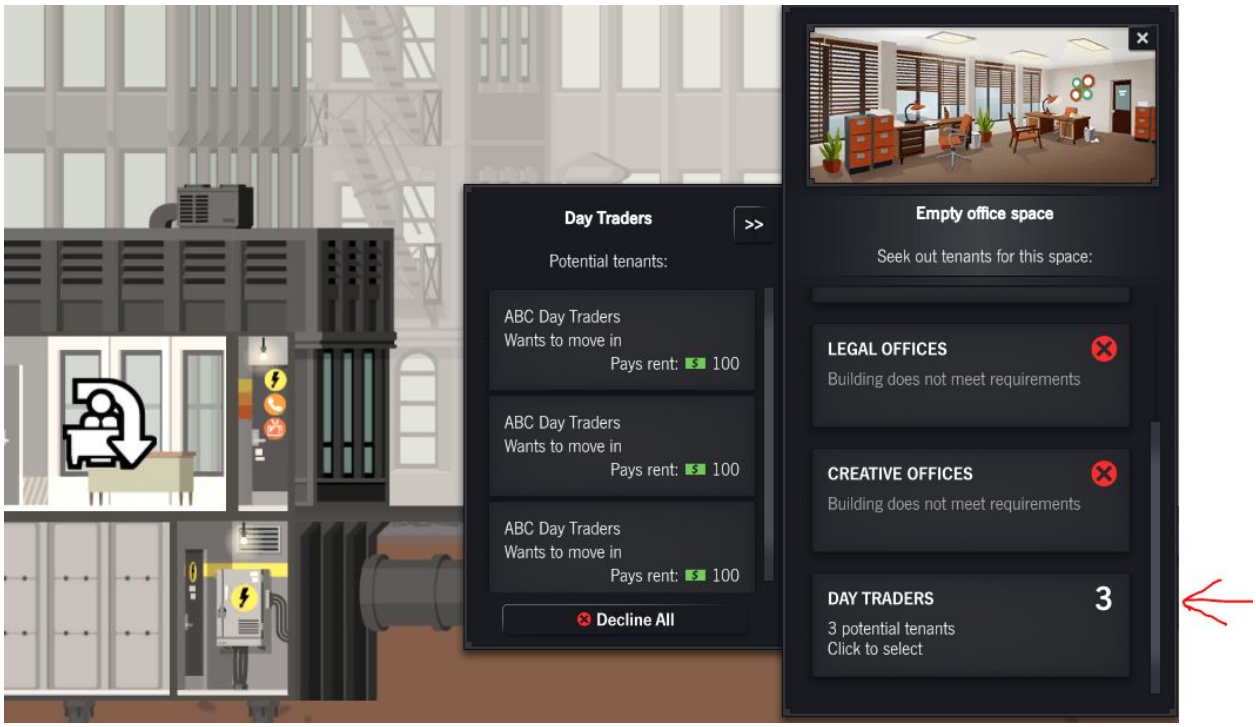
3. Now switch over to the **Description.txt** file, and replace the templated strings there too:



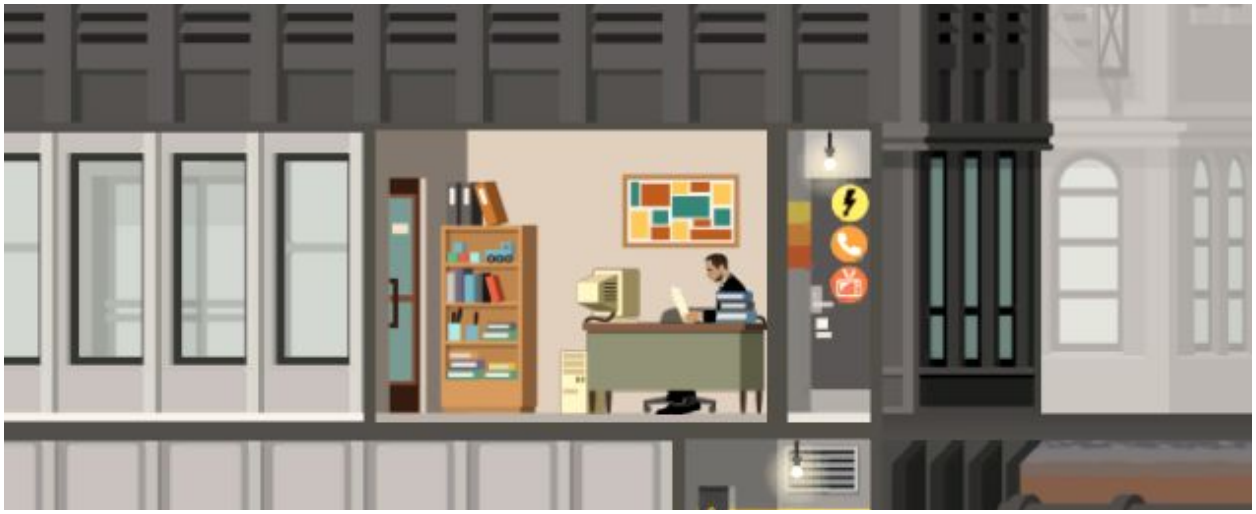
Now if we just run the game, we should see it in action. First enable the new mod:



Second, start a new game, and place down an empty office. If you click on it and scroll all the way down the list of tenant types, you should see our new Day Traders section:



And when you move one of them in, it should be using the new images:

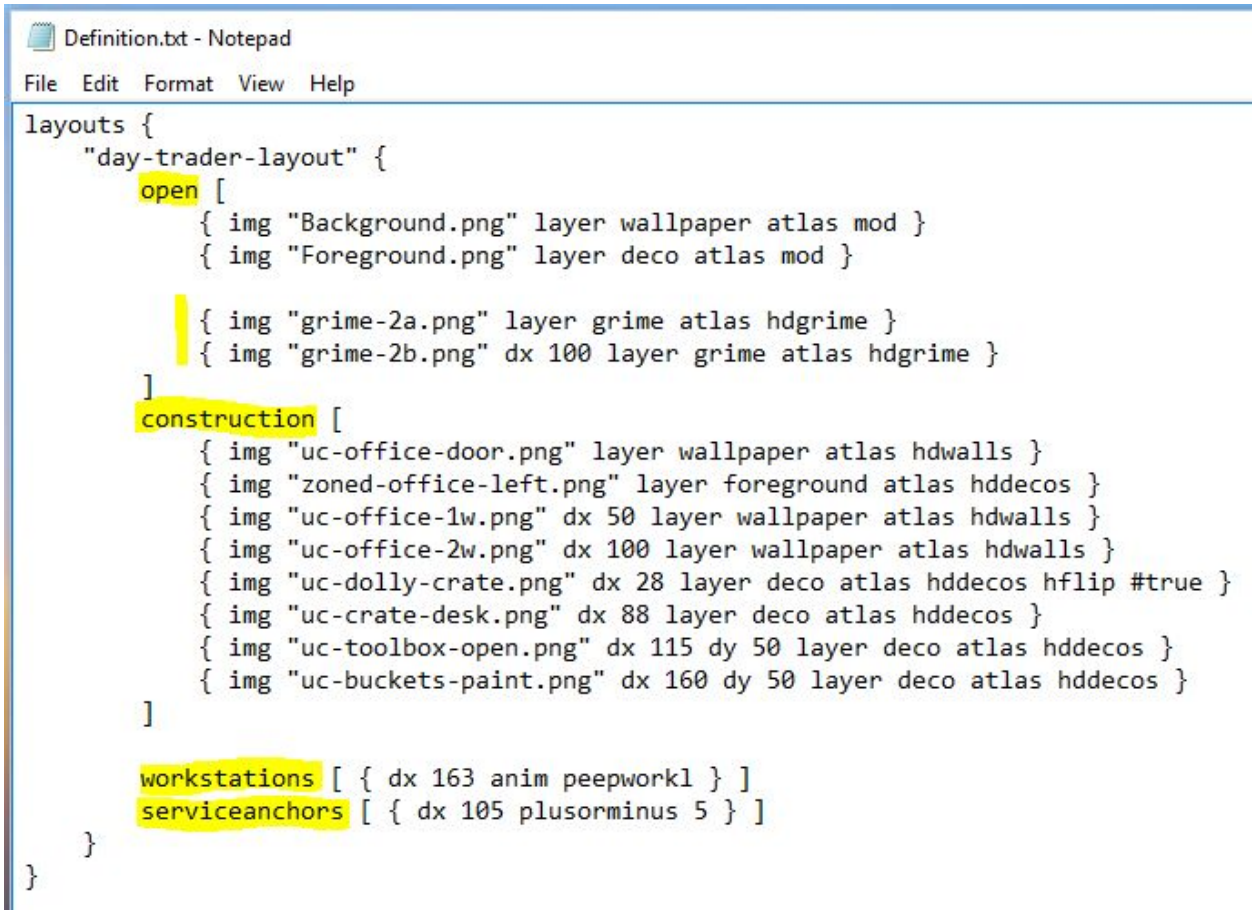


Here’s a funny thing: the sitting animation doesn’t actually match our images! The chair in the image is facing towards the camera, but the worker is sitting facing left and not actually on it.

We’ll going to fix this next. It’s also a good first task, to motivate us to look inside our new **Definition.txt** file, and see what it’s made of.

Step 2. Understanding and modifying layouts

First, open your Definition.txt and scroll down to the “layouts” section. You should see something like this:



What does all of this mean?

- **open** section is what we’ve seen before: these are the default images to be used when showing this entity. In addition to our Background and Foreground custom images that we’ve seen before, there are also two entries for “grime” images. These are overlay images that fade in slowly, if the office falls into disrepair. You should leave them in place.

- **construction** section contains a lot of built-in images for a construction site, where the builder would be finishing up construction of this entity. All of those images are built-in, but you can also change them around, or add your mod images as well.

- **workstations** and **serviceanchors** lists specify *anchoring points* for people:

- **workstations** is a list of points where workers go to work. This one is at **dx 193** (ie. 286 px from the left). Once at their workstation spot, the worker play a specific animation, in this case peepworkl. The set of valid animations is:
    - **peepworkl** means work while facing left
    - **peepworkr** means work while facing right
    - **peepworkb** means work with the back to the camera
    - **peepworkf** means work while facing the camera

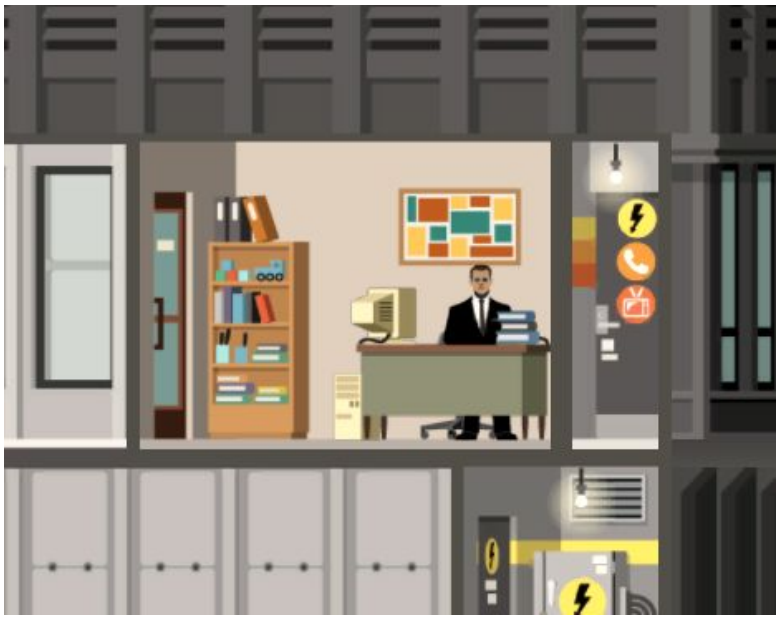
- **serviceanchors** is a list of points where service people go to make deliveries. In this case it's at **dx 105** (ie. 210 px from the left).

I think you can see our predicament here: the work station tells the worker to "work while facing left", but the image shows a chair that's facing to the camera.

Let's change the workstation definition as follows:

```
workstations [ { dx 163 anim peepworkf } ]
serviceanchors [ { dx 105 plusorminus 5 } ]
```

… and then try it again in the game:



Ahh, yes, much better!

HOMEWORK 1:

- Change workstation **dx** to some other numbers and see what happens
- Change the **anim** from **peepworkf** to one of the other valid ones (see above) and see what happens

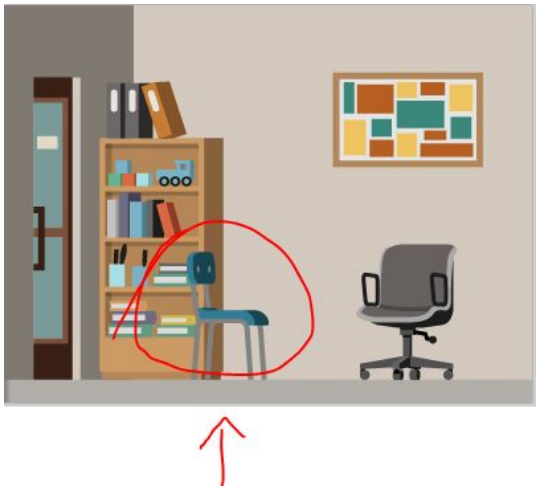
Step 3. Modifying work locations

In this and the next few sections we'll try a couple of different worker modifications. First, we'll try changing the worker's work anchor and animation, then we'll add another worker, and finally we'll change both work hours and workers to different ones.

For the first change, let's modify the worker's work location: instead of sitting behind the desk, they will sit in a chair right next to the desk.

Load up the **data / Background.png** file in an image editor, and then find a chair image to put in there as well. You can grab one from the sample images - or you can draw your own. :)

I'm going to use a modest blue office chair. Try to get something that looks roughly like this:



Background.png

Now go back to **Definition.txt** and change the work anchor:

```
workstations [ { dx 95 anim peepworkr } ]
serviceanchors [ { dx 105 plusorminus 5 } ]
```

And run the game again. Hey, now our office worker sits in the new chair to work! And by work, it looks like mainly reading the newspaper:

If the worker is slightly to the left or to the right, try adjusting the dx value until they line up with the chair.

If you make changes to the files while the game is running, the changes ***will not*** *get reloaded while the game is running.* But you don't have to quit the game completely - just quit to the main menu. Then you can update your mod, and then load up a save file or start a new game - the changes will show up.

Step 4. Adding a worker and learning about worker definitions

Our day trading office only has one person working there. How does the game know that *only one* person should work there, and how does it know *which* person? We certainly didn't tell it that! :)

The answer lies a little higher up in the **Definition.txt** file. Look for a subsection named "workplace" - it should look like this:

```
ident { template "day-trader" parent "office-level-one-base" }
sprite { layouts [ "day-trader-layout" ] }
workplace {
    workers 1
    workertemplates [ "npc-office-office.a-office.1" ]
    workhours { from 8 to 16.5 }
}
```

This section defines the workplace parameters of this office:

- **workers**: how many people work here
- **workertemplates**: which particular character types work there, and
- **workhours**: when this office is open (using a 24-hour clock, 16.5 = 16:30 = 4:30pm)

Let's add another person to the office. This will happen in two steps.

1. Change the **workplace** parameters as follows:

```
workplace {
    workers 2
    workertemplates [ "npc-office-office.a-office.1" "npc-office-office.b-office.5" ]
    workhours { from 8 to 19 }
}
```

(NOTE: Because of potential text encoding problems, do not copy/paste from this PDF - dashes will not copy properly!)

Everything here should be self-explanatory except for the second **workertemplates** entry. How do we know what the valid worker templates are? To that end, we made a handy cheat sheet. :) Check it out:
https://storage.googleapis.com/highrise-modding/docs/npc-definitions-office.pdf

Also, if you look inside that chart, you will see that each worker template already has its own work hours:

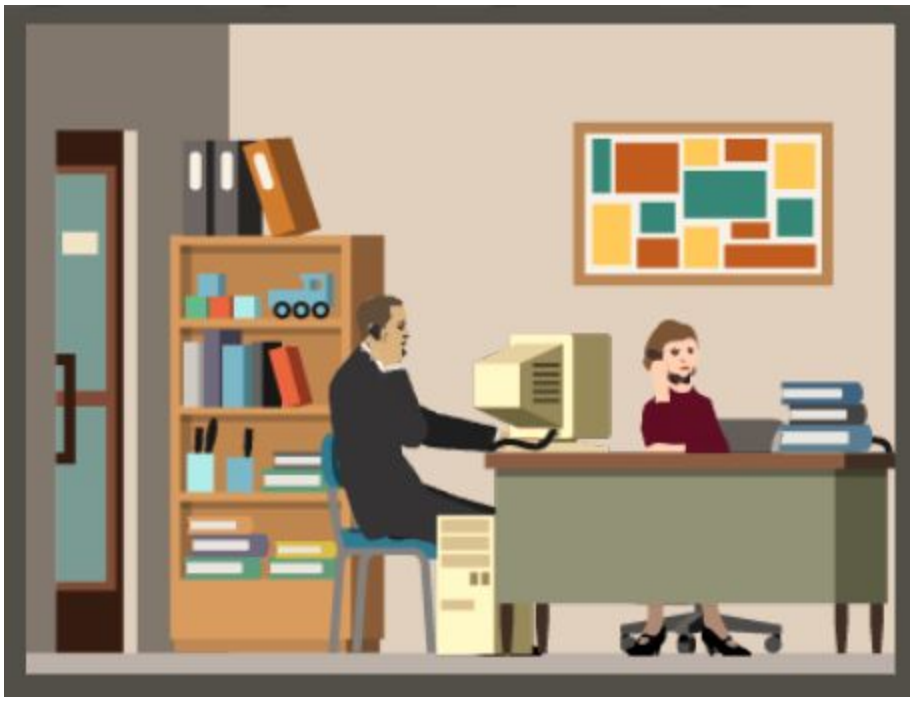
1. npc-office-office.a-office.1 work hours are 8:00 - 16:30
2. npc-office-office.b-office.5 work hours are 10:30 - 19:00

This is why we need to adjust the **workhours** as well, so that the office value matches the NPCs' work hours. (It doesn't have to match, but that could be confusing for players.)

2. Now go back to the **layouts** section, and change the work anchors.
Since we have two workers, we'll need two work places for them!

```
workstations [ { dx 95 anim peepworkr } { dx 150 anim peepworkf } ]
serviceanchors [ { dx 105 plusorminus 5 } ]
```

Now save and start the game. When you create the new office, it should have two people working there!



By the way, each worker is randomly generated, so you could end up with two male or two female workers instead - similarly for skin and hair color, they are randomized as well.

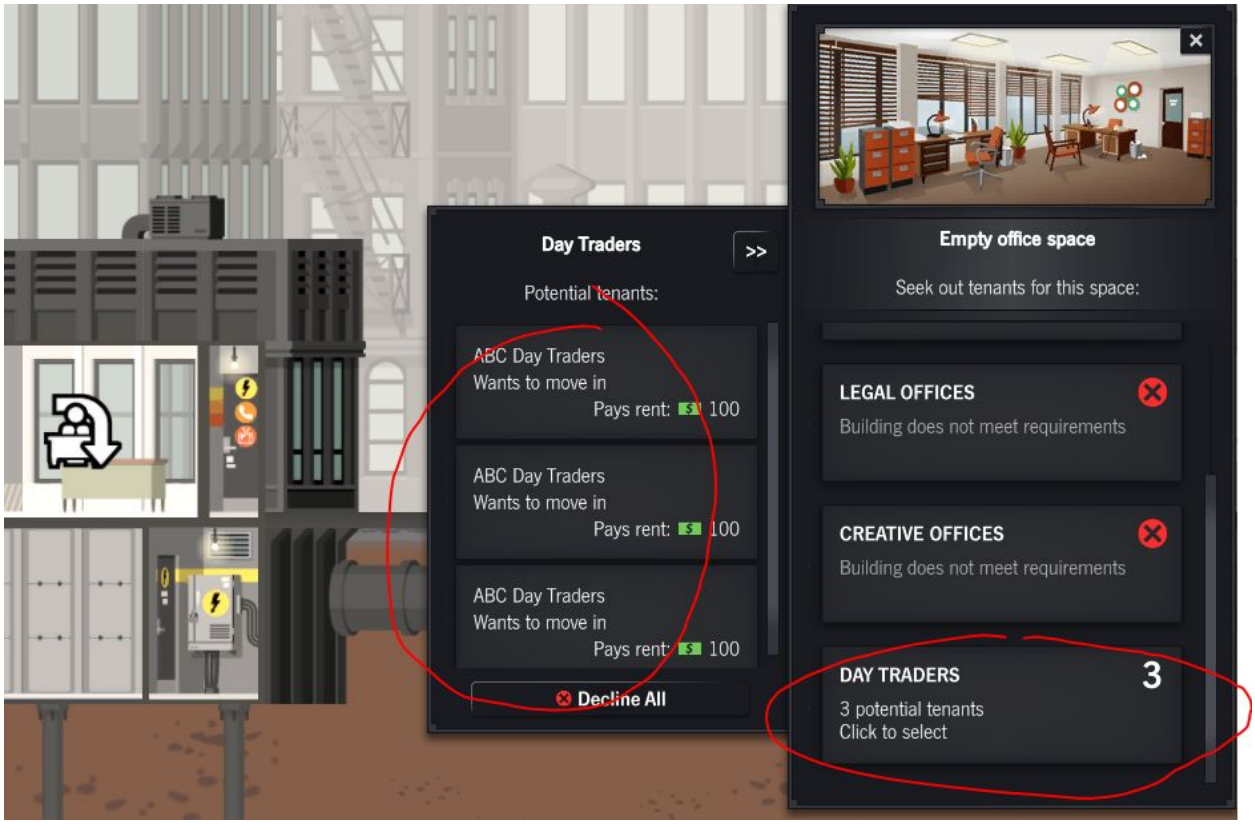
If you wait until the end of the day, you should see one person go home around 4:30pm, and the newly added one go home around 7pm.

HOMEWORK 2.

- Change your workers to some other ones listed in that table.
- Now add another chair to the background image, and add a third worker to your office.
- Don't forget to re-type the dashes in the template names to make sure they show up!

Step 5. Tenant wait lists, and wait list names

Let's go back to our mod. Remember when you placed down a spot for an empty office, and the new Day Traders category showed up there? How did that happen exactly?



The answer is: in your **Definition.txt** file there's a separate section called "movein" which describes the wait list for this kind of an office size, and how the new mod office fits in.

Here's what it looks like in your mod:

```
movein
{
    id "office-4-1"
    entries [ {
        id "day-trader-movein"
        locname "day-trader-movein.name"
        tags [ "day-trader-tag" ]
        cost { value -100 }
    } ]
}
```

The interesting parts are:

- The **movein id** (**office-4-1**) is a built-in name. No need to change this - each different template for offices, restaurants, etc will specify the right movein id for that type.
- Under **entries** we only have one entry, with the following data:
    - Entry id **day-trader-movein** is a new ID specifically for this mod. Each mod will have its own entry id.
    - Locname **day-trader-movein.name** points to a text string in the “text” section, that’s how we get “DAY TRADERS” in the list of people wanting to rent this space
    - Tags list contains a new tag **day-trader-tag** which is also made up just for this mod, and that is how the wait list hooks up to the entity definition above
    - **cost** section lists how much it costs to run an advertisement for this category

And as we saw in a previous section, there is a “text” section that contains the actual human-readable name for this movein list, as well as for the office itself:

```
text {
    en {
        "day-trader.name" "ABC Day Traders"
        "day-trader.desc" "Stock trading office, buys low and sells high"

        "day-trader-movein.name"    "Day Traders"
    }
}
```

In fact, let’s change the name of the move-in category, and also add a variety of individual business names. Change this section as follows:

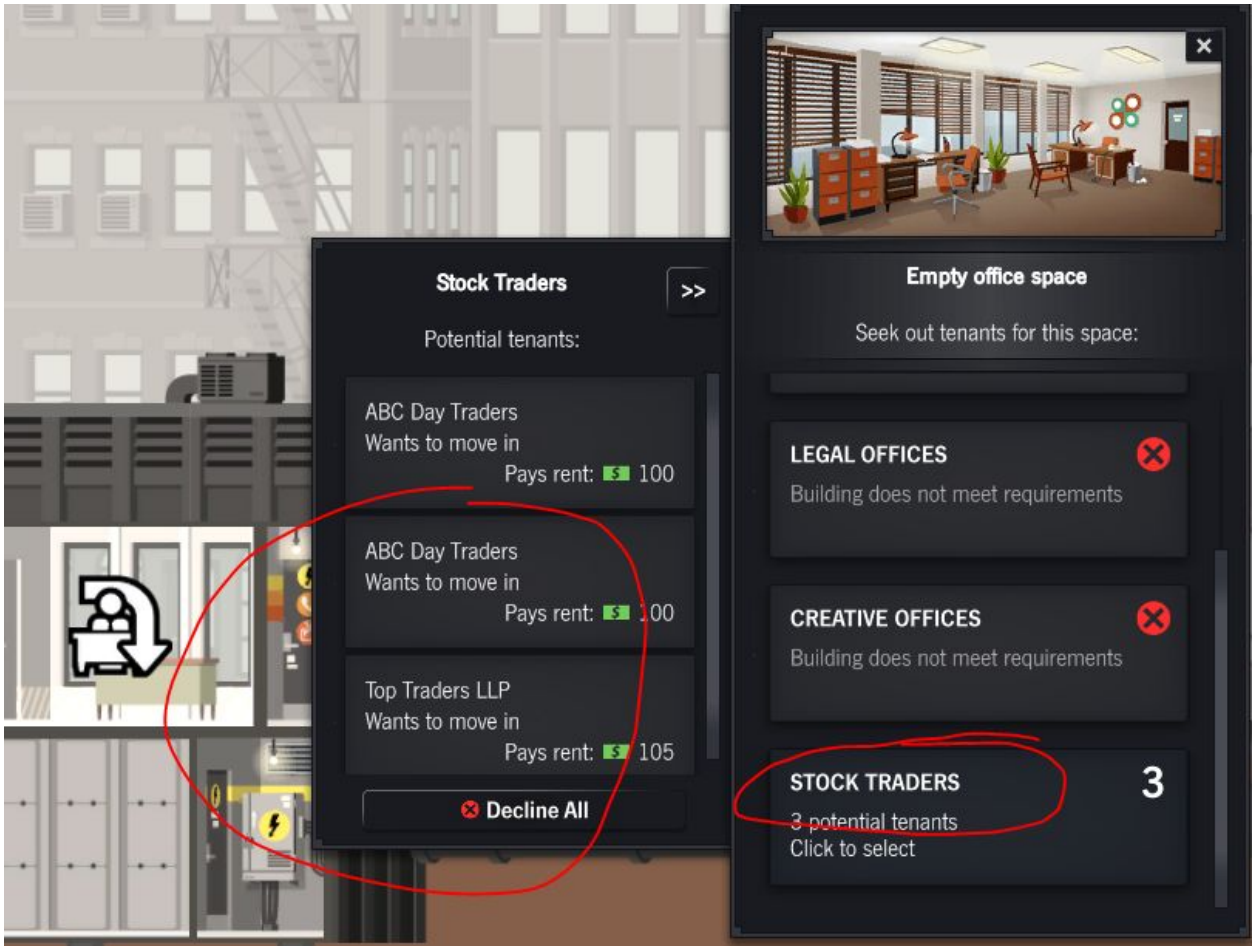
```
text {
    en {
        "day-trader.name" [ "ABC Day Traders" "Hot Trades" "Bull Market Inc." "Top Traders LLP" ]
        "day-trader.desc" "Stock trading office, buys low and sells high"

        "day-trader-movein.name"    "Stock Traders"
    }
}
```

Nota bene: see the square brackets [ ] highlighted with arrows? Don’t miss those!

Square brackets are used to define array of elements, and without commas, like this:

[element1 element2 element3 …]

In the text section, if the system sees an array of strings rather than a single string, it will pick one of them at random and use that. Randomizing names creates a nice variety of tenants.



As you can see, sometimes the system selects the same string randomly. It's easier to avoid if there are many strings to choose from.

HOMEWORK 3:

- Add more interesting office names to the array.

Step 6. Changing rents and build costs

For the last part in this modding tutorial we're going to change how much it costs to build out office space for this tenant, and how much money we're going to get in rent.

This information is at the top of your **Definition.txt** file, in a section named "unit". Look for data that looks like this:

```
unit {
    locname "day-trader.name"
    locdesc "day-trader.desc"
    loccat  "day-trader.desc"
    buildcost { value -100 mods [
        { #type heatmap-modifier map floorHeatmap value -50 }
        { #type golden-age-modifier id goldenage-offices-promoted multiplier 0.5 }
    ] }
    pays { value 100 mods [
        { #type game-difficulty-modifier hardmultiplier 0.75 easymultiplier 1.5 }
        { #type economy-modifier multiplier 20 }
    ] }
    satisfaction { value 0 mods [
        { #type rent-deal-modifier }
        { #type utilities-modifier type utilpower value -1 }
        { #type employee-happiness-modifier multiplier 0.75 onlyunhappy #true }
        { #type golden-age-modifier id goldenage-offices-satisfaction delta 0.2 }
    ] }
    tags [ office "day-trader-tag" ]
    favrestaurants [ rest-foodcourt ]
    needutilities [ { type utilpower value 1 } ]
    needservices [
```

There's a lot of *stuff* in there, but the two important elements we want are:

- **buildcost** which specifies how much it costs to build out office space of this type, and
- **pays** which is the daily rent that's being paid

Each of those has a "value" entry. **buildcost** entry has **value -100,** and the negative number means we will *take* $100 from the player when this unit is built. Meanwhile the **pays** entry has **value 100** which means this office will *give* the player $100 of rent every day at midnight.

Finally both of these entries have arrays of modifiers (or **mods**) which we don't have to worry about, but just for the sake of completeness let's look at them quickly:

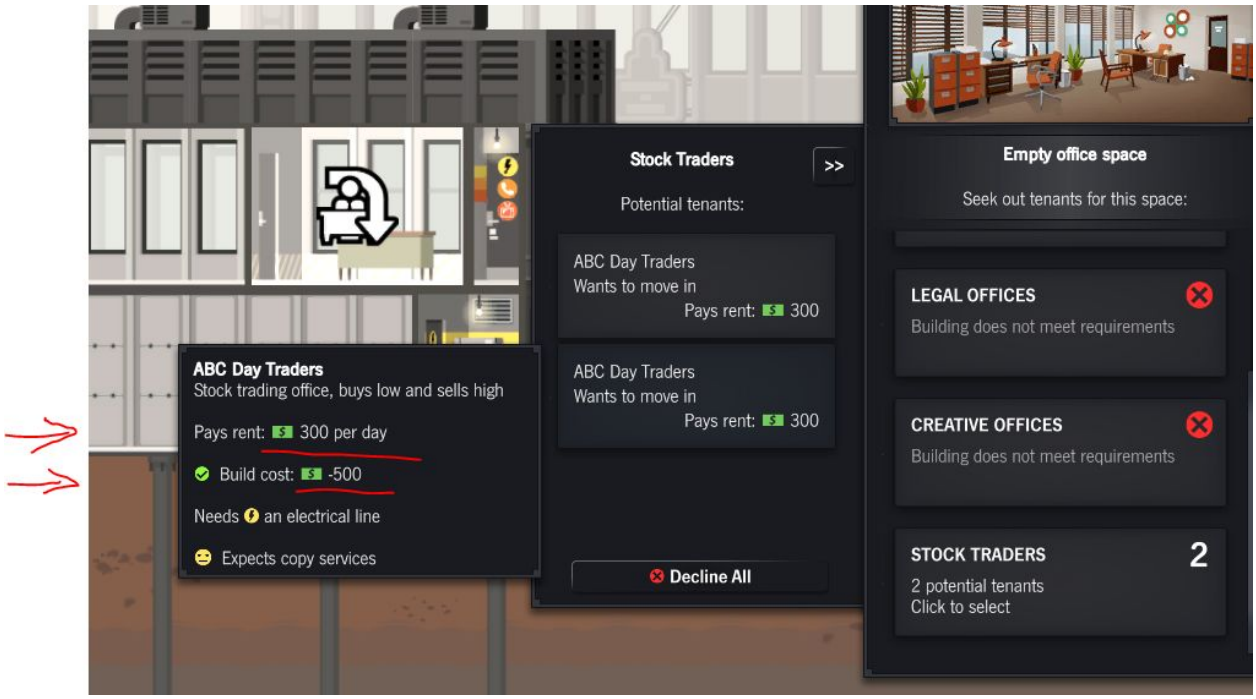
- *heatmap-modifier* changes the value based on the select heatmap
- *golden-age-modifier* changes the value when the named media campaign is active
- *game-difficulty-modifier* changes the value depending on the game difficulty level

- *economy-modifier* changes the value based on the current state of world economy
- and so on and so forth.

Perhaps in a future tutorial we will have a more comprehensive guide to all of the modifiers. For now, let's just increase both the base construction value, and the base rents that they will pay:

```
buildcost { value -500 mods [
    { #type heatmap-modifier map floorHeatmap value -50 }
    { #type golden-age-modifier id goldenage-offices-promoted multiplier 0.5 }
] }
pays { value 300 mods [
    { #type game-difficulty-modifier hardmultiplier 0.75 easymultiplier 1.5 }
    { #type economy-modifier multiplier 20 }
] }
```

(Note: the build cost must always be negative or zero, otherwise the game will throw errors! Rent must also always be positive or zero, it cannot be negative.)



HOMEWORK 4.

- Change rents to be wildly profitable, and change the build cost to be expensive.
- Now try playtesting in hard mode, and in easy mode. See if the build costs and rents are affected.

Appendix. Learning about built-in offices etc.

It's not just mods that are defined using text files. All office, restaurant, and other tenant information lives in text files inside your game's directory. The format is *slightly* different than the one for mod files: specifically, a mod puts everything inside a single file, while the game spreads its contents across several files. But nevertheless, they may be interesting to check out as a source of inspiration or to learn more about what's possible:

To find them, open the folder where the game is installed. By default, Steam will put the game in a location like **C:\Program Files (x86)\Steam\SteamApps\common\Project Highrise .** Once you're there, open the sub-folders **Game_Data / StreamingAssets**.

In there, you can see a file called **Entities.scm**. Open it in a text editor, and search for the definition of "office-level-one-a" - that's the small insurance office. Here's an excerpt of what you'll find:

```
{
    ident { template "office-level-one-a" parent "office-level-one-base" }
    sfx {
        emitter "AMB_Office"
    }
    sprite {
        layouts [ office-level-one-a ]
    }
    workplace {
        workers 1
        workertemplates [ "npc-office-office.a-office.1" ]
        workhours { from 8 to 16.5 }
    }
    unit {
        locpattern "#pattern-insurance"
        locdesc "office.one.a.desc"
        loccat  "office.one.a.cat"
        buildcost { value -100 mods [
            { #type heatmap-modifier map floorHeatmap value -50 }
            { #type golden-age-modifier id goldenage-offices-promoted multiplier 0.5 }
            ] }
        pays { value 100 mods [
            { #type game-difficulty-modifier hardmultiplier 0.75 easymultiplier 1.5 }
            { #type economy-modifier multiplier 20 }
        ] }
```

Some of this data will probably be intuitive as you read it (you've seen "buildcost" or "pays" values, for example), but you don't need to understand all of them. Just know that they're there, and that your own mods can draw inspiration from them.

Another important file you may want to look into is called **Layouts.scm**, and it defines all the layouts for all of the offices, restaurants, apartments, and other units in the building. If you search for the layout named **office-level-one-a** you should find this:

```
office-level-one-a {
    open [
        { img "office-1a.png" layer wallpaper atlas hdcom }
        { img "chair-desk-1.png" layer deco dx 158  atlas hddecos }
        { img "office-1b-left.png" layer foreground atlas hddecos }
        { img "grime-2a.png" layer grime atlas hdgrime }
        { img "grime-2b.png" dx 100 layer grime atlas hdgrime }
    ]
    construction [
        { img "uc-office-door.png" layer wallpaper atlas hdwalls }
        { img "zoned-office-left.png" layer foreground atlas hddecos }
        { img "uc-office-1w.png" dx 50 layer wallpaper atlas hdwalls }
        { img "uc-office-2w.png" dx 100 layer wallpaper atlas hdwalls }
        { img "uc-dolly-crate.png" dx 28 layer deco atlas hddecos hflip #true }
        { img "uc-crate-desk.png" dx 88 layer deco atlas hddecos }
        { img "uc-toolbox-open.png" dx 115 dy 50 layer deco atlas hddecos }
        { img "uc-buckets-paint.png" dx 160 dy 50 layer deco atlas hddecos }
      ]

    workstations [ { dx 163 anim peepworkl } ]
    serviceanchors [ { dx 105 plusorminus 5 } ]
}
```

You can see it contains a list of images that get used when the office is “open” (ie. ready) and when it’s under construction. This is where we the small office template got its layout from.

Finally there’s a third important file there called **Settings.scm**. We’re not going to describe it at all here, and just let you check it out and see what’s there.

Enjoy!