

# The Beginner's Guide to Contributing Vaults to Dungeon Crawl: Stone Soup

MainiacJoe

July, 2023

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>	6.1	Run the Python Script . . . . .	9
1.1	Purpose . . . . .	2	6.2	Using the Python Script . . . . .	9
1.2	Minimum Procedure? That Table of Contents is <i>long</i> ! . . . . .	2	6.2.1	Colors . . . . .	9
1.3	I'm already a contributor, what I can I get from this? . . . . .	2	6.2.2	Glyphs . . . . .	9
<b>2</b>	<b>What is a VAULT FILE?</b>	<b>2</b>	6.2.3	The Map . . . . .	10
2.1	Suggested Vault Types . . . . .	2	<b>7</b>	<b>Write the Vault File Header</b>	<b>10</b>
2.2	Principles of Arrival Vault Design . . . . .	3	7.1	Anatomy of a Vault File . . . . .	10
<b>3</b>	<b>Workflow Overview</b>	<b>3</b>	7.2	Name your Vault . . . . .	10
3.1	Three One-Time Tasks . . . . .	3	7.3	The Vault File Header . . . . .	11
3.2	Five Steps for Each New Vault . . . . .	3	7.3.1	Designating the Vault Type . . . . .	11
<b>4</b>	<b>One-Time Tasks</b>	<b>3</b>	7.3.2	Instructions for the Level Generator . . . . .	11
4.1	Install Offline Crawl Locally . . . . .	3	7.3.3	Customization . . . . .	11
4.2	Create a FORK of Crawl at Github . . . . .	4	7.4	The Completed Vault Code . . . . .	12
4.3	Create a BRANCH for Submitting Your Vault . . . . .	5	7.5	Doing All the Things . . . . .	12
<b>5</b>	<b>Draw a Vault Map!</b>	<b>5</b>	<b>8</b>	<b>Test Your Vault</b>	<b>13</b>
5.1	Build the Map at Piskel Online Image Editor . . . . .	5	8.1	Editing Offline Crawl's Data Files . . . . .	13
5.1.1	Starting a Piskel Sprite . . . . .	5	8.1.1	Where are the Vault Files? . . . . .	13
5.1.2	Load the Custom Palette . . . . .	6	8.1.2	Make the Actual Edit . . . . .	13
5.1.3	Guide to the Custom Palette . . . . .	6	8.2	Seeing Your Vault in Action . . . . .	13
5.1.4	Drawing Your Map . . . . .	7	8.2.1	Testing Your Vault via Wizmode . . . . .	13
5.1.5	Our Arrival Vault as an Image . . . . .	7	8.2.2	Tweaking Your Vault . . . . .	14
5.2	Saving Piskel Image Files . . . . .	8	<b>9</b>	<b>Crawl at Github</b>	<b>15</b>
5.3	Export Image from Piskel . . . . .	9	9.1	Prepare Your Vault for Submission . . . . .	15
<b>6</b>	<b>Convert the Image into Vault Syntax</b>	<b>9</b>	9.2	Move Your Vault to Your Vault Submissions Branch . . . . .	15
			9.3	"Commit" Your Edit . . . . .	16
			9.4	Make a PULL REQUEST . . . . .	17
			9.5	(Optional) Syncing with <code>crawl/crawl</code> . . . . .	18
			<b>10</b>	<b>Now We Wait</b>	<b>18</b>

# 1 Introduction

This guide teaches a minimum procedure for players who want to contribute vaults to Crawl for the first time.

- You don't need to know how to code in C, Lua, or anything else.
- You don't need to know how to use Git or Github. There are a few things that need to be done at Github, but you'll get instructions that treat it as a black box.
- You don't have to be an expert player of Crawl.

## 1.1 Purpose

My impression is that setting up the glyphs that make up the map and the submission process are the biggest hurdles for new vault designers. I wrote a Python script to help me draw my maps in an image editor, and I describe here the least complicated way to contribute.

I have enjoyed contributing and would like for other to be able to enjoy that, too. Most of my contributions have been Temples and overflow altar vaults. My most impactful contribution was probably changing the Artificer's starting skills and equipment to not box them into short blades.

## 1.2 Minimum Procedure? That Table of Contents is *long*!

Yeah, it is. Partly this is because I'm pretty thorough with explanations and instructions. But part of it is that even the most basic method of contributing is nontrivial. Please don't be intimidated! All the effort is worth it the first time you encounter your own vault in the game.

## 1.3 I'm already a contributor, what I can I get from this?

You will probably appreciate Sections 5 and 6 which explain my method for converting pixel graphics into vault map glyphs.

# 2 What is a VAULT FILE?

Crawl uses an algorithm to create levels randomly, making rooms and corridors, placing items and monsters, etc. In addition, Crawl places pre-designed terrain that it reads from a text file. This file is called a VAULT FILE and has a .des extension. Vault files serve several purposes:

- Vaults ensure that game-essential content is placed, e.g. an exit from the Dungeon, portals between branches, and a Temple.
- Vaults improve the gameplay and aesthetics beyond what the random level building algorithm can provide.
- Vaults place fun optional content such as uniques, shops, special items, etc.

Most vault files contain many individual vaults, but some vaults are complicated enough to have their own file.

## 2.1 Suggested Vault Types

These five vault types are good for beginners because they must not or need not involve placing items or monsters, which it takes experience to balance well.

**Arrival** These place the entrance to the Dungeon. Monsters strongly discouraged.

**Decor** These are placed as terrain to break up the monotony of randomly generated levels.

**Overflow Altar** These place altars to 1 to 6 gods that weren't selected to be in the Temple.

**Faded Altar** These place a faded altar

**Ecumenical Temple:** These place altars to 6 to 21 gods<sup>1</sup>. Monsters forbidden.

In this tutorial we will make an arrival vault.

---

<sup>1</sup>You can make rare Temples with fewer than 6 altars, or Temples that contain Ignis, Jiyva, or Lugonu altars. Consult `dat/des/branches/temple.des` in the last section of the file for details.

## 2.2 Principles of Arrival Vault Design

Arrival vaults contain the entrance to the Dungeon, the spot where each game begins (and hopefully ends). Arrival vaults are always welcome. It is the one thing that every game will have no matter how quickly a YASD comes, so when the most frequent feature have more variety it reduces repetition. There are several arrival vaults that I've seen many times.

Guidelines for designing arrival vaults are found in the offline version's file structure at `crawl-ref/docs/develop/levels/guid` or click [here](#) to see it at Github.

The two most relevant guidelines for this demonstration arrival vault are these:

- Try to come up with small maps.
- Arrival vaults should have multiple entry points, escape hatches, or enough space to permit tactics.

We are going to make a oval of transparent rock around the Dungeon entrance, all in the middle of an oval room. We will put some statues inside for decoration.

## 3 Workflow Overview

This section serves as an outline of the instructions that follow.

### 3.1 Three One-Time Tasks

There are two things you need to do first.

1. Create a local copy of latest release of Crawl
2. Create a online copy of Crawl at Github
3. Create a space within your Github online copy that you will submit vaults from

### 3.2 Five Steps for Each New Vault

These are the steps in my method for making vaults. You'll do these for each vault you make.

1. Create a picture of your vault using a specific pixel art website, [piskelapp.com](http://piskelapp.com).
2. Convert the picture into text using a Python script designed to work with Piskel's output.
3. Test your vault using wizmode on your local copy of Crawl.
4. Edit your vault into your online copy of Crawl on Github.
5. Ask the Dev Team to add your vault to official Crawl.

## 4 One-Time Tasks

### 4.1 Install Offline Crawl Locally

In order to test vaults, you will need an offline installation of Crawl on your local machine. Tiles vs console will affect how your vaults look to the player, but not how the vault file is processed by the Crawl executable. Go ahead and choose the version you're used to.

1. Browse to [crawl.develz.org/download.htm](http://crawl.develz.org/download.htm).
2. Download the latest stable version.

**Windows:** Download the zips. Extract the folder, and everything you need will be in that extracted folder.

**MacOS:** **STUB**

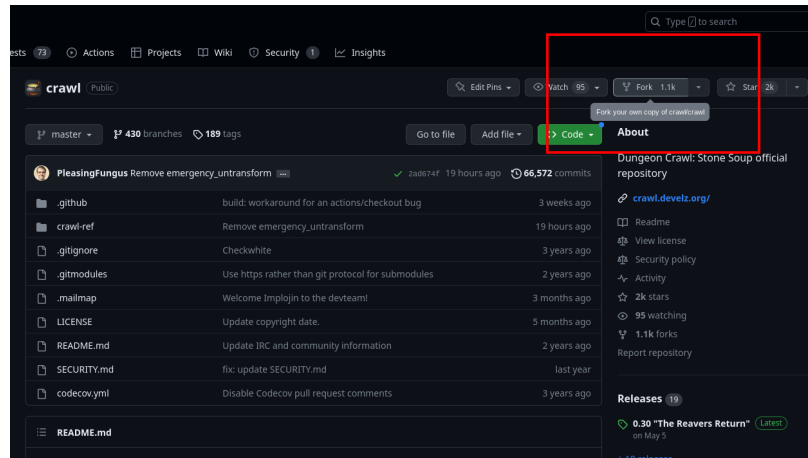
**Linux:** Consult <http://crawl.develz.org/download.htm#linux>.

## 4.2 Create a FORK of Crawl at Github

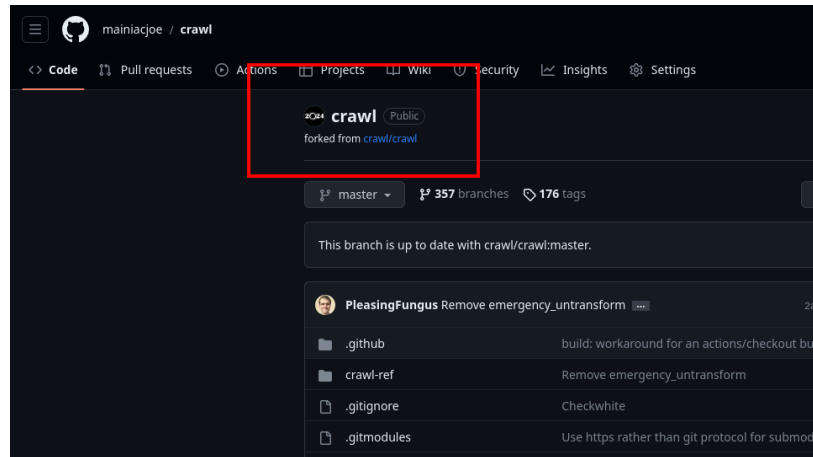
Eventually you'll need to upload your vault from your local machine that you created it on. You will be making the same edits in online Crawl as you did in your offline Crawl, but you don't have edit privileges on the official Crawl code. So you need a personal online copy of Crawl that you can edit. In Git-speak this copy is called a FORK.

1. Log in at [github.com](https://github.com)
2. Browse to [github.com/crawl/crawl](https://github.com/crawl/crawl)
3. Click Fork in the upper right. See Figure 1.

[github.com/your\\_username/crawl](https://github.com/your_username/crawl) is created, identical to what the latest and greatest Crawl was at that moment<sup>2</sup>. You ought to be sent to your fork automatically. Figure 1 has screenshots. Vault file syntax is changed very rarely, so the differences between your fork of trunk Crawl and your local copy of the latest released version are almost certainly immaterial.



(a) The Fork button is in the upper right at [github.com/crawl/crawl](https://github.com/crawl/crawl)



(b) Look for your avatar and forked from [crawl/crawl](https://github.com/crawl/crawl).

Figure 1: Creating your fork of Crawl at Github

<sup>2</sup>If you wait a while, eventually there will be new things added to Crawl and your fork will be out of date. You can ignore this, but if you want to stay in sync with Crawl trunk, see Section 9.5.

### 4.3 Create a BRANCH for Submitting Your Vault

Your online copy of Crawl at Github is called your fork. Each submission you make to Crawl is kept separate by putting it in its own BRANCH. A branch segregates a set of changes from the rest of your fork, so it can be discarded if you decide you don't want those changes without having to manually undo all the edits.

The first two panes of Figure 2 show where to click to get to the **Create new branch** popup. Your source needs to be `your_username/crawl` and the branch this new branch comes from needs to be `master`.

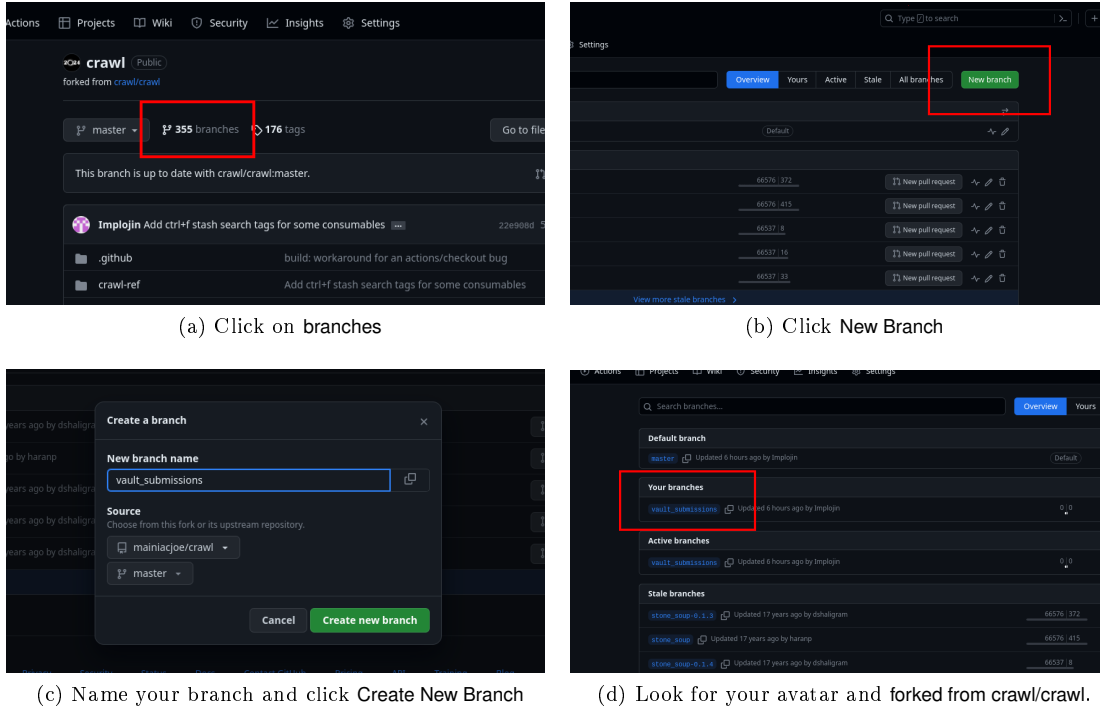


Figure 2: Creating a branch at Github for vault submissions

## 5 Draw a Vault Map!

### 5.1 Build the Map at Piskel Online Image Editor

Piskel is an online pixel art editor that can export images as text files. It is optimized for creating sprite sheets and animated gifs, but it can make single frame images just fine. Here's what you need to do.

1. Upload and select a custom palette configured to work with the text file parsing script.
2. Draw your vault map with the colored pixels. The palette colors are keyed to common glyphs that make up the ASCII map of the vault file.
3. Export the image from Piskel as a `.c` file.
4. Use a custom Python script to translate the `.c` file into the vault file's map syntax.

I chose Piskel because it is easy to download an image as a text file of pixel colors. This file has a format that is easy to read with Python.

#### 5.1.1 Starting a Piskel Sprite

Go to [www.piskelapp.com](http://www.piskelapp.com) and "Create Sprite".

### 5.1.2 Load the Custom Palette

The Palette Tool is in the lower right. Select the yellow plus icon, “Create a new palette”. Import the file, DCS Vault Builder.gpl, then **Save**. Figure 3 has screenshots. With cookies you ought need do this only once.

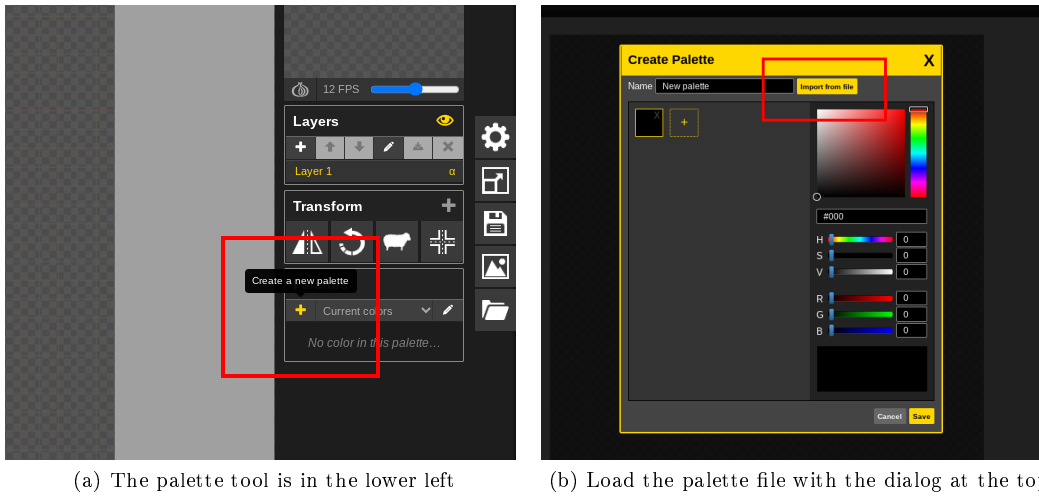


Figure 3: Loading the custom palette into Piskel

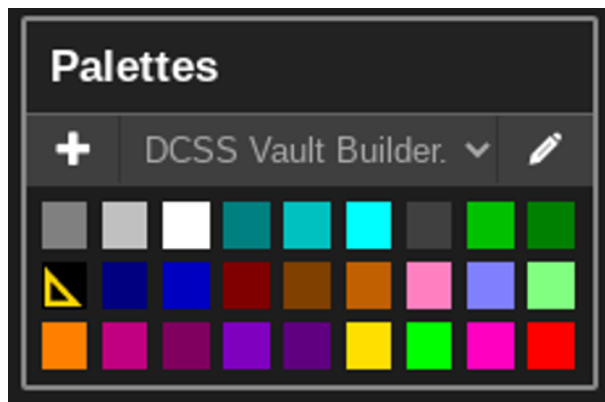


Figure 4: The custom palette for Piskel




























### 5.1.3 Guide to the Custom Palette

The Python script you’ll use later translates pixel color into vault map glyphs. At Table 1 you’ll see:

- The RGB value of the pixel color
- The pixel color
- The default vault map glyph it is translated to
- The vault map name for the glyph
- The color name that the Python script uses

You will also be able to redefine color-to-glyph mapping for greater flexibility.

Table 1: Color to Glyph Mapping

808080		x	Opaque Rock Wall	Dark Gray
c0c0c0		c	Opaque Stone Wall	Gray
ffffff		X	Opaque Permawall	White
008080		m	Transparent Rock Wall	Dark Cyan
00c0c0		n	Transparent Stone Wall	Cyan
00ffff		o	Transparent Permawall	Light Cyan
404040		v	Metal Wall	Charcoal
00c000		b	Green Crystal Wall	Green
008000		t	Tree	Dark Green
000000		.	Rock Floor	Black
000080		w	Deep Water	Dark Blue
0000c0		W	Shallow Water	Blue
800000		l	Lava	Dark Red
804000		+	Normal Door	Dark Brown
c06000		=	Runed Door	Brown
ff80c0		G	Granite Statue	Pink
8080ff		T	Water Fountain	Pale Blue
80ff80		B	Altar	Pale Green
ff8000		@	Entry Point	Orange
c00080		{	Stairs Up	Plum
800060		<	Hatch Up	Dark Plum
8000c0		}	Stairs Down	Purple
600080		>	Hatch Down	Dark Purple
ffdf00		\$	Gold	Gold
00ff00		P		Bright Green
ff00c0		Q		Bright Pink
ff0000		R		Bright Red

#### 5.1.4 Drawing Your Map

Piskel is similar to other pixel editors. Consult the table and draw your walls and floors and features using the appropriate colors. If you want something that isn't in the table, use the last three colors which don't map to anything. If you need more than three, re-use a different color for a feature you aren't using, or make a custom color. You'll get a chance later to specify which glyph you want for each color; you aren't locked into the table.

#### 5.1.5 Our Arrival Vault as an Image

Figure 5 is our arrival vault as an image in Piskel. It has a Plum upstairs, eight Pink statues, and Dark Cyan transparent rock walls. We give the level builder more flexibility in how it attaches the arrival vault to the rest of D:1 by not specifying a rock wall around the perimeter. The transparent pixels around the edges are not a part of the vault map; the level generator can put whatever it wants there.

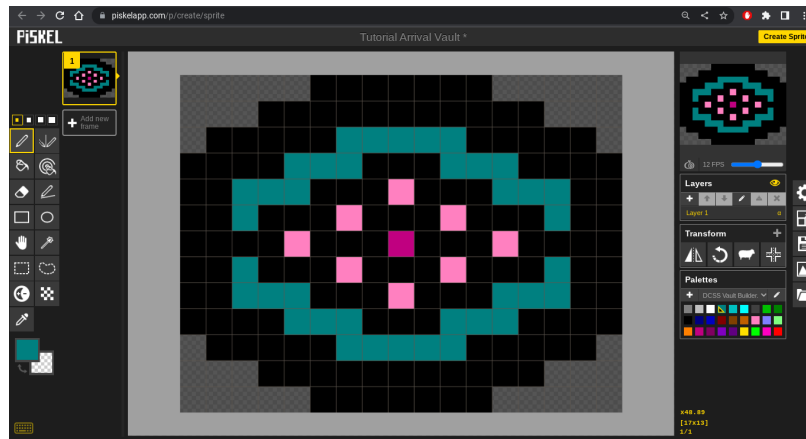


Figure 5: Our finished vault map image at Piskel

## 5.2 Saving Piskel Image Files

Piskel used to have cloud storage, but discontinued it. Instead, you can save Piskel projects to your local machine as a file with a `.piskel` extension. This is not saving an image; see the next subsection.

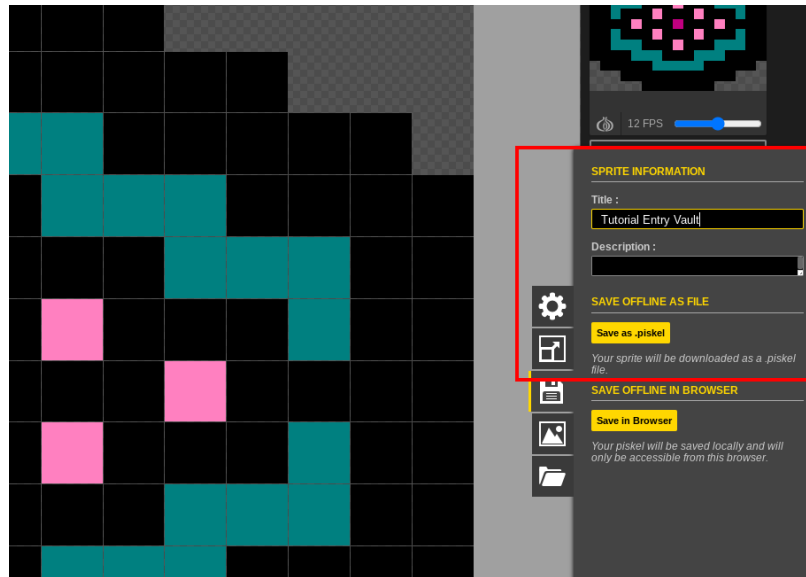


Figure 6: Saving our project as a `.piskel` file



## 5.3 Export Image from Piskel

The “Export” button on the right looks like a picture of a mountain and the Sun. Select the **Others** tab and **Download C file**. We’ll not learn more about how to use Piskel here. If you prefer a different image editor, use the **Import** button on the right, then export your image as a C file. The Python script is written to parse C files from Piskel, and may not work on other editors’ exported C files, so importing an image to Piskel and exporting its C file ensures that it can be converted into map glyphs.

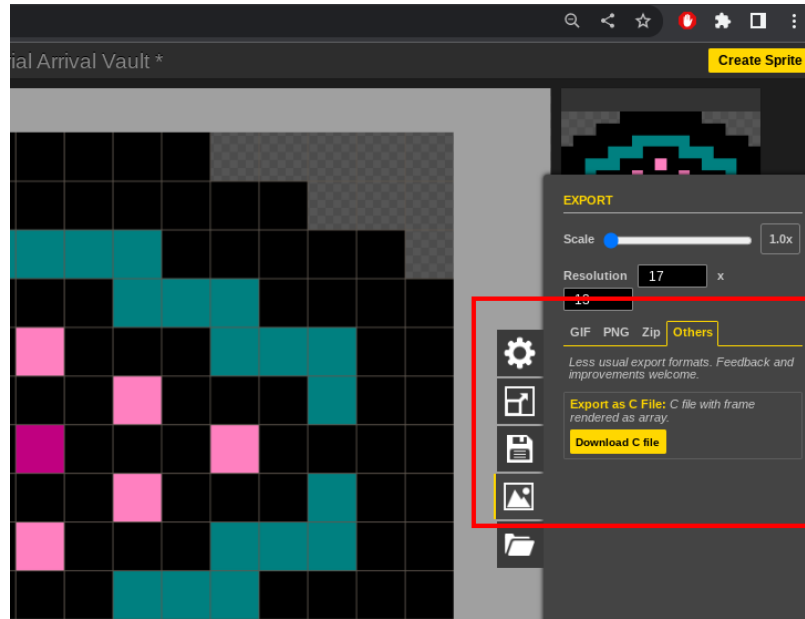


Figure 7: Exporting your image from Piskel

## 6 Convert the Image into Vault Syntax

### 6.1 Run the Python Script

The Python script is named, `vaultmap.py`. It parses the C file from Piskel and converts each pixel into a glyph according to Table 1. The script is designed to be run from the command line, but you can also run it from your favorite Python IDE. It’s written in Python 3.

Make sure the exported C file is in the same directory as `vaultmap.py`. The syntax for executing the script from the command line is: `python3 vaultmap.py filename`. The filename is case-sensitive. If your filename has spaces, use quote marks around it. The `.c` extension is optional. If the filename can’t be found, or if you run the script from an IDE, you will be prompted for a filename.

### 6.2 Using the Python Script

#### 6.2.1 Colors

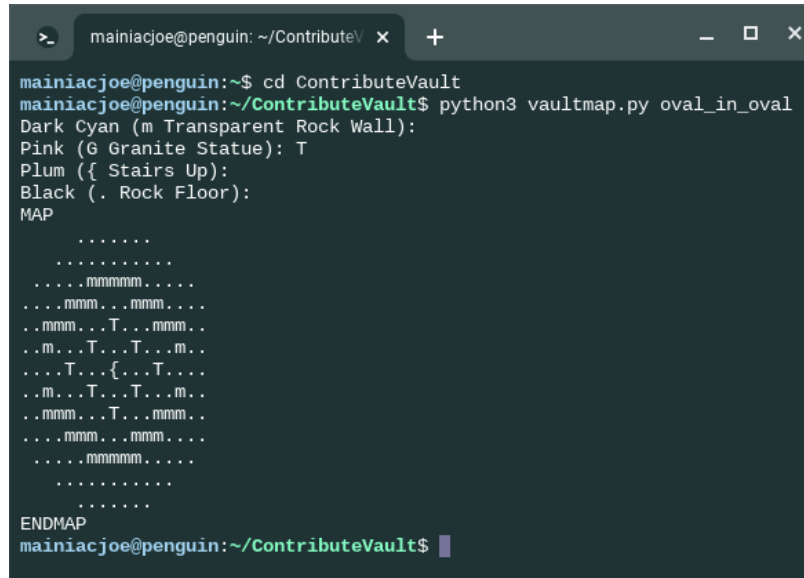
The script will detect all the colors in your vault map image. If it is a color from Table 1, it will identify it with the name from that table. If the color isn’t found in Table 1, it will identify it by its hex RGB.

#### 6.2.2 Glyphs

It will prompt you for the glyph you intend each color to represent. Accept the given default glyph with `[Enter]`, or enter a different glyph. Transparent pixels though are always translated into a space with no input from you. To provide an example of this we’ll change Pink from statue to fountain.

### 6.2.3 The Map

When you've assigned a glyph to each color in the image, the script will print out the map in the terminal window. Copy the output from **MAP** to **ENDMAP** inclusive, and paste it into any text editor. There isn't much that can be done about how pixels (and tiles) are square and glyphs are rectangular.



```
mainiacjoe@penguin: ~/ContributeV x +
mainiacjoe@penguin:~$ cd ContributeVault
mainiacjoe@penguin:~/ContributeVault$ python3 vaultmap.py oval_in_oval
Dark Cyan (m Transparent Rock Wall): T
Pink (G Granite Statue): T
Plum ({ Stairs Up): {
Black (. Rock Floor): .
MAP
.....
.....
.....mmmmmm.....
...mmm...mmm...
..mmm...T...mmm..
..m...T...T...m..
...T...{...T...
..m...T...T...m..
..mmm...T...mmm..
...mmm...mmm...
.....mmmmmm.....
.....
ENDMAP
mainiacjoe@penguin:~/ContributeVault$
```

Figure 8: Exporting your image from Piskel

## 7 Write the Vault File Header

### 7.1 Anatomy of a Vault File

The ASCII map itself is only part of the vault. Every vault's code must have the following required elements:

- **NAME:** *the name of the vault*
- **MAP**
- *the ASCII glyphs that make the map*
- **ENDMAP**

**NAME:** and **ENDMAP** delineate the beginning and end of each individual vault in a vault file. Most **.des** files have many individual vaults in them.

### 7.2 Name your Vault

You already have the last three required elements from the Python script. On the line before **MAP**, type **NAME:** and the vault name. Vault names cannot have spaces within them, but there must be at least one space between it and the colon in **NAME:**. Here and for the rest of the document, *nick* means your username or nickname, and *vaultname* means your name for the vault itself.

**Arrival** **NAME:** *nick\_arrival\_vaultname*

**Decor** **NAME:** *nick\_vaultname*

**Overflow Altar** **NAME:** *nick\_overflow\_vaultname*

**Faded Altar** **NAME:** *nick\_ecumenical\_vaultname*

**Ecumenical Temple** **NAME:** *nick\_temple\_vaultname*

## 7.3 The Vault File Header

The next step is to write the (technically optional) header lines. The header tells the level making algorithm what it needs to know about your vault to place it accurately. You can skip this section and come back to it later. If you do, make sure your vault file looks like Figure ?? before moving on to Section 8.

### 7.3.1 Designating the Vault Type

After the name, enter in a line that tells the level generator the role your vault has in the game. This line always starts with TAGS: but what goes after it depends on the type of vault you are making.

**Arrival** TAGS: `arrival`

**Decor** TAGS: `extra decor`

**Overflow** TAGS: `temple_overflow_generic_#` where # is the number of altars in the overflow vault<sup>3</sup>.

**Temple** TAGS: `temple_altars_#` where # is the number of altars on the Temple map.<sup>4</sup> You also need a separate line, **PLACE**: `Temple`

**Faded Altar** There is no specific tag that identifies this type of vault, rather put `: ecumenical_altar_setup(_G)` on the line immediately before **MAP**. The colon is part of this command.

### 7.3.2 Instructions for the Level Generator

These next two lines tell the level generator how it can place your map.

**Orientation** **ORIENT**: `float` tells the level generator that you don't care where the vault is placed on the map or whether it gets flipped or rotated.

**Transparency** TAGS: `transparent` tells the level generator that it can use any empty space on the edge of the vault map for other things.

Next include one or both of these two lines.

**Monsters** TAGS: `no_monster_gen` prevents monsters from being randomly placed in your vault. This is mandatory for Temple vaults and strongly encouraged for arrival vaults.<sup>5</sup>

**Water** TAGS: `no_pool_fixup` will prevent deep water from being turned randomly into shallow water. If you don't place any deep water in your vault or you do but don't mind if some of it is shallow, you can omit this.

### 7.3.3 Customization

Even with hundreds of arrival vaults, sooner or later you'll see an arrival vault more than once. The vault code includes many ways to randomize the vault so that it stays fresh even with several views.

For our purposes here will restrict ourselves to a very basic customization. We will give the level builder several options for the transparent walls and water fountains (since we changed Pink from G to T when we read the C file).

**The Walls** The point of the walls being transparent is that the player can see through them. So our substitutions need to be something that the player can also see through or over. **SUBST**: `m : m#lw` tells the level generator to take every instance of m in our vault file and replace them all with m (no net replacement), #, l, or w. Consulting Table 1, this means that our transparent walls might become an iron grate, lava, or deep water instead. You probably noticed that # is not in the table. We need to tell the level generator that we want this glyph to mean iron grate, with the line **KFEAT**: `# = iron_grate`. Now if the level generator chooses # to replace m with, it will not throw an error.

<sup>3</sup>You can designate the altars in the vault to be specifically to a certain god. Consult `dat/des/altar/overflow.des` for details

<sup>4</sup>It is possible to make Temples and overflow altar vaults with a variable number of altars. The more altar counts your map has, the better the chance that you will encounter your own Temple or overflow in the game. How to do this is beyond the scope of this tutorial, but inspecting the vault files in `dat/des/branches/temple.des` and `dat/des/altar/overflow.des` and consulting the vault building documentation should get you off to a good start.

<sup>5</sup>Note that this does not prevent monster placement defined by the vault file itself. We'll see an example of this later with plants, or you might want to make a vault with some fearsome monster behind unbreakable glass to greet foolish adventurers when they enter the Dungeon.

**The Fountains** These are just decoration, so we'll let them maybe turn into statues, orcish idols, green crystal walls, trees, fungi, plants, or bushes with this line: `SUBST: T : TGbt123V`. Here we also need an extra line, because fungus, plant, and bush—but not tree—are technically monsters. This is the extra line we need: `MONS: fungus, plant, bush`. The numeral glyphs are used to specify monsters in a vault. We need to declare what each numeral represents, as the index of an array of monster names. The first element has an index of 1.

## 7.4 The Completed Vault Code

We're done. This is our completed vault file. Even though I introduced tags above one line at a time, you can put them together separated by spaces on one line. Any amount of spaces in the header lines is considered as one space by the parser, so we can make it look readable with no penalty.

Spaces in the map glyphs, though, have a special meaning: “not a part of the vault”. The `transparent` tag tells the level builder that it can put whatever it wants there to integrate your vault into the rest of the map.

Confirm that your vault file looks like Figure ?? before continuing to Section 8.

```
NAME:    mainiacjoe_arrival_oval_in_oval
TAGS:    arrival transparent
TAGS:    no_monster_gen no_pool_fixup
ORIENT:  float
SUBST:   m : m#lw
KFEAT:   # = iron_grate
SUBST:   T : TGbt123V
MONS:    fungus, plant, bush
MAP
        .....
        .....
        .....mmmmmm.....
        ....mmm...mmm....
        ..mmm...T...mmm..
        ..m...T...T...m..
        ....T...{...T....
        ..m...T...T...m..
        ..mmm...T...mmm..
        ....mmm...mmm....
        .....mmmmmm.....
        .....
        .....
ENDMAP
```

Figure 9: The finished vault code

## 7.5 Doing All the Things

Much more information about writing vaults can be found in the offline folder structure at `crawl-ref/docs/develop/levels`, or click to [see it at GitHub](#). Looking at vaults that others have written will be instructive also.

## 8 Test Your Vault

You will test your vault by pasting it into your offline copy of Crawl and inspecting it with wizmode.

### 8.1 Editing Offline Crawl's Data Files

#### 8.1.1 Where are the Vault Files?

Vault files are in sub-folders of `dat/des`. Depending on your OS, this folder's location is:

**Windows Zip:** `crawl-ref/source/dat/des` is in the extracted folder.

**MacOS:** **STUB**

**Linux:** `/usr/share/crawl/dat/des`

For the five suggested vault types, these are where they are within `/dat/des`.

**Arrival Vaults:** `arrival/simple.des` will be where your vault probably belongs. Consult the arrival vault guidelines at `/docs/develop/levels/guidelines.md`, or click [here](#) to see it at Github.

**Decor Vaults:** `variable/float.des` These vaults must have `ORIENT: float` in their header.

**Overflow Altar Vaults:** `altar/overflow.des`

**Faded Altar Vaults:** `altar/ecumenical.des`

**Ecumenical Temple Vaults:** `branches/temple.des`

These vault files typically have a comment section at the beginning with tips or requirements for making the vault.

#### 8.1.2 Make the Actual Edit

In our example we are making a small arrival vault with very few bells and whistles. Adding your arrival vault code to Crawl is surprisingly straightforward:

1. Close any instances of offline Crawl that you have running.
2. Open `dat/des/arrival/simple.des` in your favorite text editor.
3. Paste your vault code at the very end of the file<sup>6</sup>.
4. Save the file.

That's it! The next time you start offline Crawl, your vault to will be in its repertoire.

### 8.2 Seeing Your Vault in Action

A problem may have occurred to you: to see your new vault, Crawl has to choose it from the hundreds of arrival vaults at its disposal when it starts your game. In order to test it, we need to force Crawl to use our vault so we can ensure it looks and behaves like we want it to. Wizmode is the feature of Crawl that lets us do this.

#### 8.2.1 Testing Your Vault via Wizmode

Wizmode is entered by starting a game, then pressing `&`. After telling the Crawl that `Yes` you meant to do that, it will ask for a Wizmode command. Type `G`, then `[Enter]`. This dismisses all monsters on the level so you can test your vault in peace. Subsequently, `&` serves as a prefix that says the next keystroke is a wizmode command. These are the steps for placing your vault:

1. The wizmode command to place a vault is `&L`, after which you enter the vault name as defined by `NAME:` in the vault code you wrote above.

---

<sup>6</sup>Other vault types, for instance Temples or overflow altars to specified gods, have a specific spot in the `.des` file where you need to paste your vault. The comment section of that `.des` file will describe this to you.

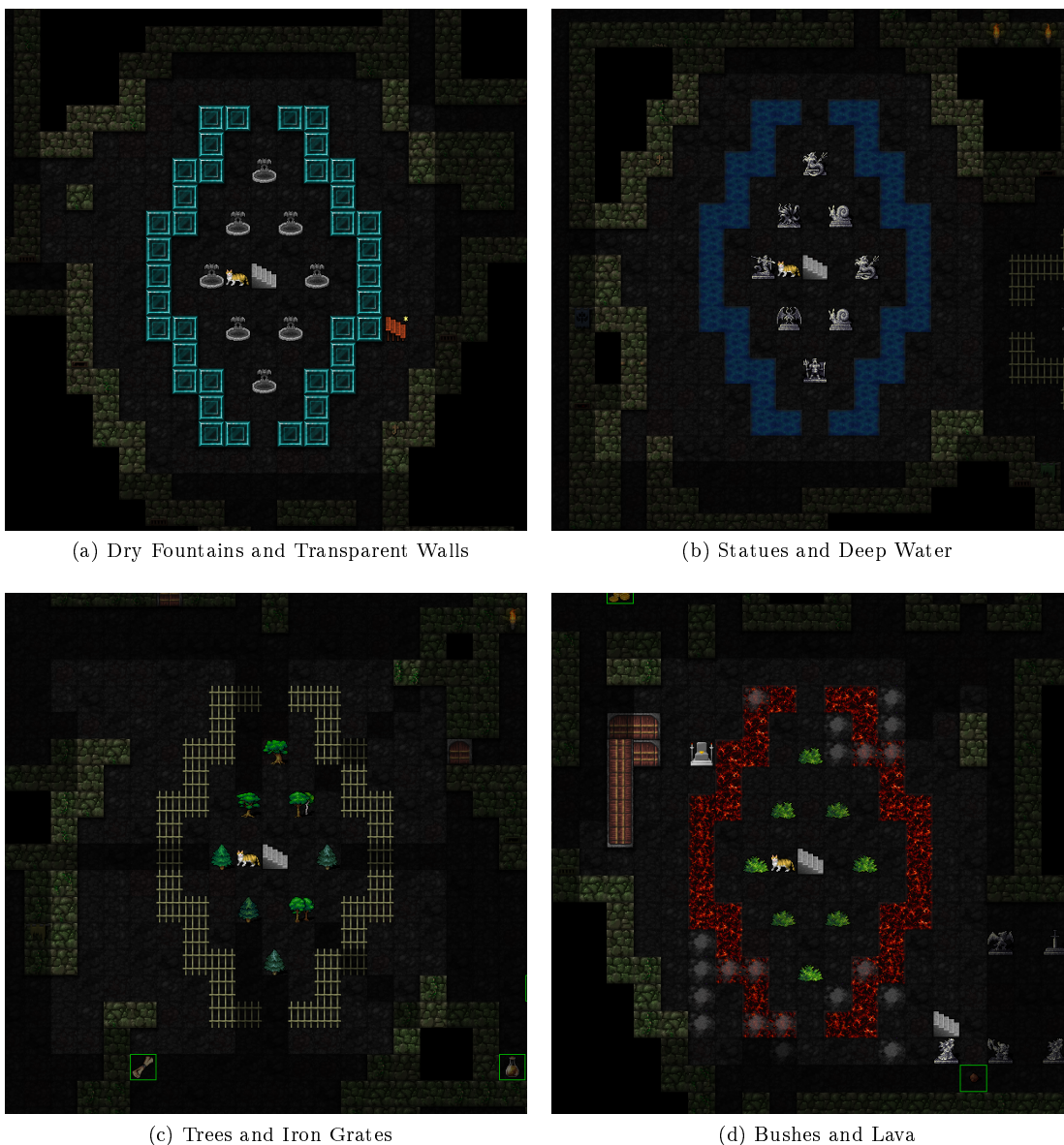


Figure 10: Various SUBST Effects

2. Use wizmode magic-mapping `&f` and then `X` to look at your vault. Note that the dungeon exit will be replaced with a simple upstairs in these placements.

Typing the name of your vault each time can be a pain. So make a macro to do it with a single keystroke: `[Ctrl] D`, then key `1`, then `&L nick_arrival_vaultname`. You can also make a macro for magic mapping: `[Ctrl] D`, then key `2`, then `&f`. Then a simple `12` will place an instance of your vault.

We set up a substitution to vary the wall type and decoration type of our entry room. Make sure you start enough games to see all your variations. With experience you'll be able to anticipate most these changes and usually not have to view multiple iterations of your vault.

### 8.2.2 Tweaking Your Vault

While wizmode lets you place your vault many times without starting a new game so you can see all your SUBST effects, if you actually edit your vault code in `simple.des` you'll need to close and restart Crawl. It only reads `.des` files during the `loading maps` message that you get when you first launch the executable. Figure 10 shows a few variations of our sample arrival vault.

In Figure 10d you can see how some of the previous terrain was overwritten. I had gone down to D:2 and done &G all again because I wanted a good screenshot and D:1 had gotten cluttered.

## 9 Crawl at Github

GitHub is a website that hosts software projects and facilitates the collaboration that makes these projects possible. Crawl is one of these projects. Earlier we made a copy of Crawl and stored it in our personal space on Github. Now will move our offline vault file edit into our Github copy of Crawl.

You may see a message that says something like, **This branch is 6 commits behind crawl:master**. You can safely ignore this, but if you want to make sure you are keeping up to date with the latest changes in Crawl trunk, see Section 9.5.

### 9.1 Prepare Your Vault for Submission

There are two things we need to do at GitHub.

### 9.2 Move Your Vault to Your Vault Submissions Branch

Once you are happy with how your vault looks in Wizmode, it's time to move it from your local computer to this new branch on your fork of Crawl at GitHub.

1. Before doing anything else, make sure that you have switched to the `vault_submissions` branch. Go to your fork's home page at [https://github.com/your\\_username/crawl](https://github.com/your_username/crawl). Click **Branches**, and then `vault_submissions` under **Your branches**. Figure 11 has screenshots.

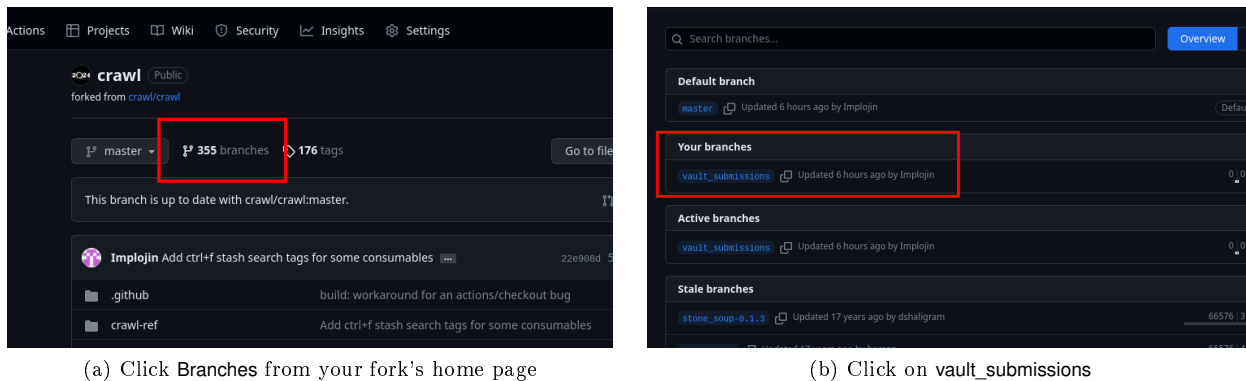


Figure 11: Switching to your `vault_submissions` branch

2. Find the same `.des` file in your fork online. The URL above will have a `crawl-ref` folder.
3. Double-click to enter folders just like exploring your local hard drive with your OS until you arrive at `crawl-ref/source/dat/des/arrival/simple.des`. It will look like Figure 12.

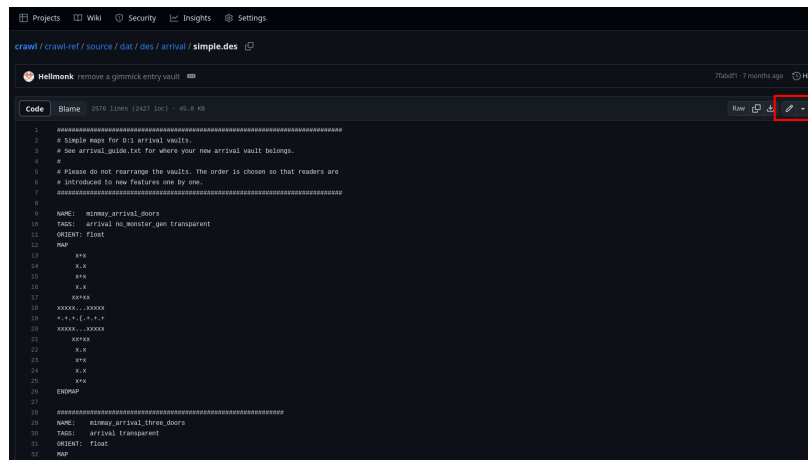


Figure 12: simple.des, view mode

4. Immediately above line 1 of the file, at the right end of the bar that gives the number of lines in the file, click the pencil icon, “Edit this file”.
5. Copy your vault code from the end of the offline instance of `simple.des` and paste it at the end of the online instance of `simple.des`.

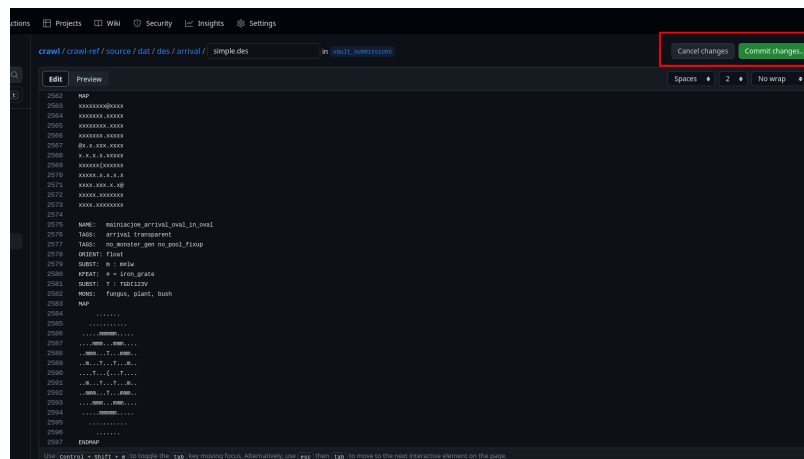


Figure 13: simple.des, edit mode with our vault pasted to the end

### 9.3 “Commit” Your Edit

If you look for a “Save” button, there isn’t one!

Because GitHub is designed to work with Git, we do have to adapt to some of its terminology even though this presentation isn’t a Git tutorial. Instead of saving individual files, with Git you take a snapshot of the entire project, which might include changes to several associated files. In Git-speak this is making a **COMMIT**. Figure 14 has screenshots.

1. At the upper right of the page, find **Commit changes**, as in Figure ??.
2. Change the upper text field to, **Add one arrival vault to simple.des**. Keep this under 50 letters.
3. Write the name of your vault in the extended description, plus anything else you want to say about it.
4. Retain the **Commit directly to the vault\_submissions** branch option, and click **Commit changes**.



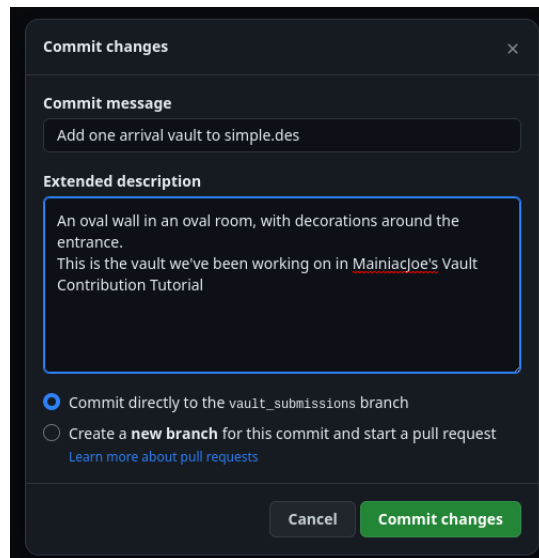
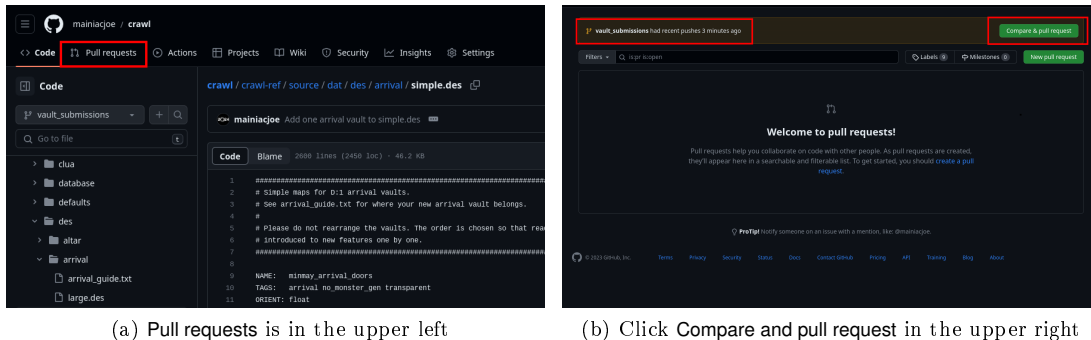


Figure 14: The Commit popup

## 9.4 Make a PULL REQUEST

You and I do not have write-privileges to official Crawl, so we made a personal fork of Crawl and edited that. Now we want to ask the Dev Team to look at our new vault, and add it to Crawl. In Git-speak, this is called a PULL REQUEST. We want to ask the Devs to pull the edit from our fork into official Crawl.

Go back to the root page for your fork, [github.com/username/crawl](https://github.com/username/crawl). Towards the top of the page, find **Contribute** then click, **Open pull request**.



(a) Pull requests is in the upper left

(b) Click Compare and pull request in the upper right

Figure 15: Pull Request

On the next page, the branch settings need to be as follows. These are likely the default settings anyway.

**base repository:** `crawl/crawl`

**base:** `master`

**head repository:** `username/crawl` That's your GitHub username.

**compare:** `vault_submissions`

Click **Create pull request** here and on the next page.

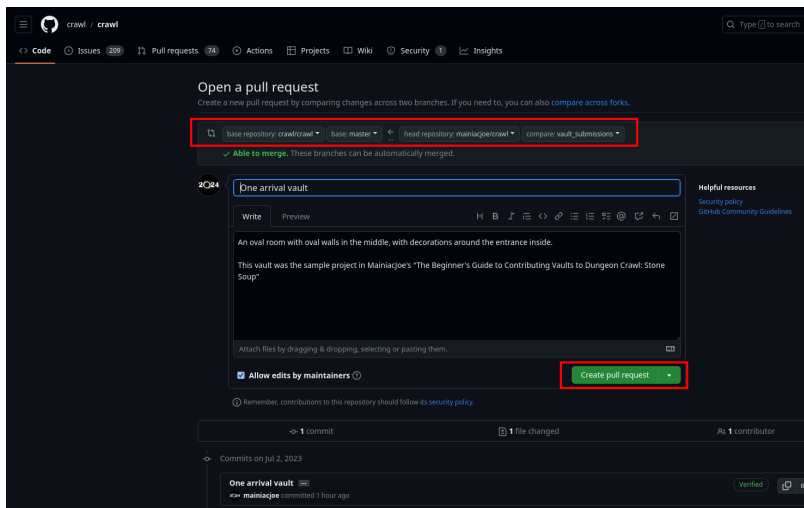
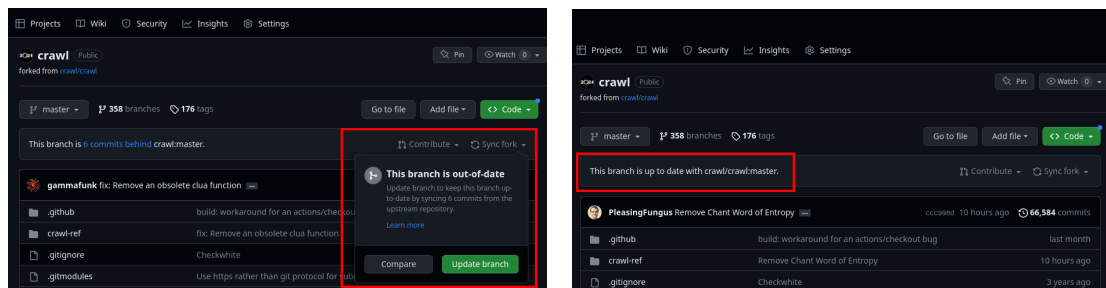


Figure 16: Confirm branch settings before creating the pull request

## 9.5 (Optional) Syncing with crawl/crawl

It's probably been a while since you first made your fork of Crawl at Github, way back in Section 4.2. It's not necessary to keep your fork of Crawl up to date with official Crawl, but you may want to do so. Figure 17 shows you how.



(a) Click on Sync Fork, then Update branch

(b) You want it to say, This branch is up to date with crawl/crawl:master

Figure 17: Syncing Your Fork with Official Crawl

## 10 Now We Wait

Our vault is now a new pull request that belongs not to our fork, but to official Crawl. You can view it at [github.com/crawl/crawl/pulls](https://github.com/crawl/crawl/pulls).

The Devs will eventually look at your vault. Crawl isn't the only thing in their lives, and even though you're excited about your first contribution, one more arrival vault isn't the highest priority.

Chances are your vault isn't quite ready to be added as is. The feedback you are likely to get will be things like tweaks to your glyph variations to avoid problems you didn't anticipate, requests to change the layout a bit to make it play more nicely with the level generator, or game balance issues. These are all things that the Devs, having a broad-picture view of how all the Crawl code fits together, aren't going to expect you to know with your first vault. They are nice people and won't make you feel stupid.