Base Building Game Documentation:

# File Locations

## World

This contains all of the stuff to do with the world, including the world generation withing the Planetary Forge section, Buildings in the Buildings section, and then the saving and loading of the world.

## Renderer

This contains all of the rendering stuff, so the drawing functions for tiles, buildings, and the general world drawing.

## Player

Hold the player class and the research, as well as some files for the legs of the player, but these are not working atm.

## Main

This contains the main running program in Program.cs, and some functions and variables that don’t really fit into separate parts.

## Images

The images are sectioned up into pretty obvious files, and the image handlers ill explain later in the **How to Add Images** section.

## Handlers

The handler has several different types of handler in it, like the one for menus, commands, the boat one, or the general one.

## Entities

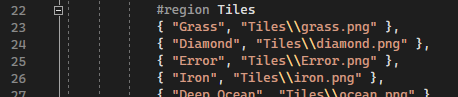
Entities hold lots of stuff, like the transporters in Men.cs, and items and other stuff like that, basically anything that isn’t tied to a tile and has the ability to move.

# How to Add Images

This is a long and annoying process soooo.

1. Add the image to the images folders in the correct place with a sensible name, this can be either a png or a bmp.
2. Go into the ImageHandlers\Image Dictionary.cs file and open the correct region in the file, if the image you are adding doesn’t seem to fit in any of those then just add a new region via doing #reigon TestName and then do #endreigon TestName at the end.

Then just add a entry like this



1. Now you should be able to draw using that image.

# Overview

So, this is where the game is gonna get broken down.

Ima break it down in order of how the player runs through it.

First it is the init screen, which should have some bars at the bottom of the screen in order to allow the player to see what the game is loading, and it will help with debugging cause then you can see what is taking time or where it is stuck. It should look kinda like the curseforge one, if you don’t know what that looks like ask M.

Then there should be a menu, and when the user presses space to start, they should be opened to a menu that says online or offline, so we can have online functionality. For the online system… idk man its kinda a bit too late, oopsie!!!, anyway for the offline system you should have the option for loading or creating, and then

## Init Screen

#Init #Initialise #Start #Load #Image #LoadSettings #LoadTexturePacks #LoadImages #LoadFancyTiles #LoadCutscenes #LoadTemplates #EntityThreadLoader #InitialisePercent

The init screen has been made woooo.

The main code is in the “Main\Initialiser.cs” file. The way it works is that you pass in functions into the InitFunctions dictionary. A computer screen shot of a program code

AI-generated content may be incorrect.

The string on the left is the text that will be displayed, and then on the right you have a class that you pass a method into. The method needs to have no inputs and no return type. You should use the InitialisePercent variable stored in General in order to let the renderer know what percentage through the proccess you are. This is what fills the bar on the left. InitialisePercent should be from 0 to 100.

The rendering for this is scary and SDL so if you really need to go into it ask M

## Main Menus

#MenuState #MenuStates #Enum #Menu

Okie so the menu system is mainly controlled by the “MenuState” variable (which is stored in the GeneralVariables.cs file) This is an enum that has a couple different optionsA screenshot of a computer

AI-generated content may be incorrect.

There is also a function above this in the Menus.cs

## Coding Style

Everything should have camel case, except for Enums, where \_’s can be used for flags, but in general keep to the system of camel case. They should all have capital letters at the start.

All classes and methods should have capital letters at the start and also use camel case. Be clear with your names pretty please 😊.

Namespaces should use \_’s rather than camel case.

All interfaces should start with an I.

I accidentally did the start of the project really badly and most of the code is in a big fat public partial static class. This is **BAD**. I’m **SORRY**. If you can fix a part of it that you are working with then please do, but for anything else that is added **DO NOT ADD IT TO THAT CLASS**.

With namespaces, if it is a very different part of the project like my world generator, then give it a separate namespace, IE Planetary\_Forge for that. Everything else should be under Base\_Building\_Game, so stuff like Base\_Building\_Game.Handlers so that different parts of the program are correctly sectioned. These parts can and should have a static main in order to allow for testing of individual components.

### Program Structure

Each program should generally be very modular, for example the men have it so that in the Run function, it looks like this.

*/\**

*plan*

*more plan*

*\*/*

*If (thing)*

*{*

*DoThing()*

*}*

*Else*

*{*

*DoOtherThing()*

*}*

This way it is easy to follow and each part can be changed ykyk you all did comp sci A-Level something something modularity.

### XML

Very recommended for any part of the program that interfaces with another part. All code should operate as a black box within its namespace, with any entrances or exits from that box having as much documentation as possible.

### Shutdown

If you want to have the game shut down at any point, Base\_Building\_Game.General.running should just get set to false. The whole game should clean itself up.

### Privacy

Try to keep everything private, but if you think it needs to be used by other parts of the program make it internal.

**NEVER USE PUBLIC PRETTY PLEASE <3**