So, you want to make Doom mods

by Agent_Ash aka Jekyll Grim Payne

There are a few things you have to know first. Doom exists in many forms, formats, and standards, and some of those are changing rapidly. Before starting your project, you need to make some right choices, otherwise there's a good chance that 6 months in you'll realize you're using some awful outdated methods.

Step 1: Choosing a source port

Most people don't play the original ("vanilla") Doom nowadays. Instead they use a source port: a program that allows running Doom on modern systems, with better graphics and also gives you access to a variety of tools and tricks that the original Doom didn't have. In most cases a source port only needs the IWAD file from the original game to work: that is, doom.wad, doom2.wad, tnt.wad or plutonia.wad (also don't forget about hexen.wad, heretic.wad, strife1.wad and chex.wad, they're also usually supported).

However, maps and mods created for one source port may not necessarily work in another, because source ports are not exactly the original game, they're just software that can use the original game's resources to "recreate" the game.

You *can* still make maps and even some mods in vanilla format: one advantage of this is that they'll always work with any source port (and some will even work in the original, MS-DOS version of Doom). The downside is that, obviously, you'll be extremely limited in what you can do.

Choosing a source port defines limitations, possibilities and tools you'll have at your disposal.

The more or less relevant source ports nowadays are:

- GZDoom — Feature-wise, the most advanced source port to date. Supports 3D geometry, dynamic lighting, 3D models in IQM (animated, rigged), MD3 (animated) and OBJ (static only) formats, materials for models and textures (normal, specular, PBR), shaders, ACS scripting, DECORATE scripting, ZScript, UDMF (universal Doom map format), and a ton of other little things. Can run .wad and .pk3 (zip) files. The major downside is that **multiplayer is p2p only, limited to 8 players**. Client/server may happen but not in the foreseeable future. The biggest drawback is that. As such, GZDoom is best suited for singleplayer projects.
    - advanced gameplay scripting (ZScript)
    - advanced map format (UDMF)
    - shaders, models and the best lighting
    - supports pk3/zip

    - doesn't support recording or playing demos (technically it does, but they'll only be playable in the specific version of GZDoom used for recording, otherwise they'll go out of sync)
    - only p2p multiplayer, 8 players max, no mid-game joining, only DM and Coop modes
- LZDoom — A fork of GZDoom with similar capabilities. The main difference is the renderer that will run on much older hardware; the downside is that for that reason it will not be able to do some of the most advanced rendering-based stuff. **Currently discontinued.** Feature-wise it's similar to GZDoom 4.6.1.
- Zandronum — a fork of ZDoom. Has fewer advanced features: doesn't support shaders, ZScript, or many of the latest DECORATE functions and features. At the same time it supports client/server functionality, so **Zandronum is usually the go-to port for multiplayer** and MP-oriented projects (it also supports up to 64 players in some modes). Features a few custom ACS scripts that aren't available in GZDoom.
    - client/server functionality, mid-game joining, up to 64 players in some MP modes
    - limited gameplay scripting (outdated DECORATE only)
    - advanced map format (UDMF)
    - models (but with fewer features), dynamic lighting
    - supports pk3/zip
- QZandronum — a fork of Zandronum aimed at improving the netcode and the general code base. Mainly aimed at supporting Quake Champions: Doom Edition but can be used for other purposes as well.
- Delta Touch — This is a combined app that brings GZDoom, LZDoom, Zandronum Chocolate Doom and PrBoom+ to smartphones. It may slightly lag behind some of the source ports but always gets updated eventually. The reason why you might need to know about it is, many people play Doom on smartphones, and if you make a mod for GZDoom, chances are it'll be incompatible with Delta Touch just because Delta Touch hasn't got some features *yet*. So, either be prepared to tell people to wait a bit, or look into the stuff you've done: perhaps you can roll back a couple of insignificant features to make your work more compatible.
- PrBoom+ — a port primarily designed to run Doom "the way it was". The gameplay is slightly more vanilla-precise in some aspects than GZDoom (a common example is Spider Mastermind being usually only killable with 2 BFG shots, while in GZDoom you can easily do it with 1 due to different handling of blockmap). Doesn't support the most modern features and mostly just looks better than vanilla (offering higher resolution and OpenGL support). It's mostly used to run vanilla-format maps on modern systems, but it *does* have its own Boom map format which is close enough to vanilla but offers more tools. Do not use it if you want advanced features.
    - good for testing/playing vanilla-styled maps
    - supports demo recording and playback
    - supports only .wad files
    - vanilla (Doom in Doom), Boom and MBF map formats only
- Vanilla Doom — as in, Doom without any source ports. Since running Doom as-is may be difficult on modern systems, Chocolate Doom is usually used, which is a

source port that runs Doom as close to the original as possible and doesn't support anything that the original Doom didn't.

Step 2: Choosing a map format and the map editor

Maps go into .wad files. That's the only universal thing for all formats. *Map* WADs, as opposed to resource/mod wads, can't be made into PK3s (more on PK3 below). Map WADs are archives arranged in a specific order that you shouldn't (and needn't) mess with.

The most current editor, regardless of the map format, is **Ultimate Doom Builder**. It can be used to create maps in any format, from vanilla to UDMF, and has a lot of quality-of-life features that make mapping easier, regardless of format.

You might have heard of some other editors, so here's a brief overview of the ones you do *not* normally needs to use:

- Doom Builder — the most well-known (but not the first) Doom map editor that UDB is based upon. Creating vanilla-format maps in it is fully covered in the well-known Doom Builder Guide, which is a good starting point for a mapper. Hasn't been in development for many years.
- Doom Builder 2 — an advanced version of DB. Hasn't been in development for many years.
- GZDoom Builder — a version of DB updated to work with GZDoom features, such as 3D floors and polyobjects. Is no longer being developed by the original author.
- GZDoom Builder Bug-Fix — a branch of GZDB with even more features and fixes. This is what was later renamed into Ultimate Doom Builder. There's one reason you might need to use it: Ultimate needs OpenGL 3.3, so if your hardware doesn't support it, you'll have to use Bug-Fix.

Apart from the OpenGL thing, the above are mentioned for historical purposes. If you started working with any of them, don't worry: switching won't be that difficult. And switching between GZDB-bf and UDB won't require any changes at all, it'll even read your old config files.

If you're on **Linux**, however, you can either run UDB via Wine, or try to compile UDB for Linux yourself, or you'll be stuck with SLADE. SLADE, however, is primarily an archive editor, not a map editor (more on that below), so it won't be very convenient.

Now, onto map formats!

When you launch UDB and hit "New Map", you'll have to choose a **Game Configuration**. Game configs in UDB define a number of things, such as which IWAD they're going to use (doom.wad, or doom2.wad, or hexen.wad, etc.), and which map objects will be available to you. You can even create your own custom configs (and you'll likely need to do that if you decide to make a huge total conversion or a standalone game). Before choosing a config, you'll have to make it available and add base resources for it by hitting F6.

The most important thing a config defines is the **map format** you're going to use. Let's cover them briefly:

- **GZDoom: UDMF** — If you want all the advanced features, such as 3D floors, ACS (level/event scripting) and whatnot, *this* is your go-to format. It's the most convenient format of all for multiple reasons (for example, giving linedefs IDs is much easier than before, and sectors can have multiple tags at once). Obviously, there are versions of UDMF for Doom, Hexen, Heretic, etc. They're the same thing, they'll just use a different iwad file as a base resource file.
    - One of the prominent differences of UDMF is its support of ACS, a scripting language originating from Hexen. It allows to script map events; it's more a mapping tool than a scripting tool and is normally covered by mapping tutorials.
- **Doom: Doom 2 (Doom format)** — Vanilla format that will create maps just like they were in Doom 2, which will run on anything starting with MS-DOS. Usually chosen primarily because of its compatibility, but it's also a stylistic choice: it allows the author to play against the limitations and see how creative they can get. It's also a good format for beginners since it uses simple tools. Do not use it if you want anything more advanced and modern than classic Doom.
- **Boom** — A map format initially designed for the Boom port (see PrBoom+ above) and supported by the vast majority of existing ports (not just ZDoom family). It still has some popularity nowadays, mostly for compatibility and simplicity (it offers many more tools than vanilla Doom, even if much fewer than GZDoom). It's traditionally often viewed as an accepted "baseline" standard due to its broad compatibility with various source ports. If you're looking for highly advanced features, however, there's no reason to start with it, and developing for it won't necessarily be easier, since UDMF offers a number of quality-of-life improvements over Boom.

That's all the map formats you need to know about. But there's one very common beginner mistake related to choosing formats, so I'll explicitly mention it:

Do not use Doom in Hexen format!

"Doom in Hexen" used to be the most advanced map format before UDMF was introduced, because it supported most of the cool new features such as 3D floors, polyobjects, ACS, and such. It is now *completely irrelevant*. You can technically still use it but you'll make mapping significantly harder for yourself (due to lack of key quality-of-life features) while not gaining anything (all ports that support Doom in Hexen *also* support UDMF, so there are no compatibility concerns whatsoever).

If you started with Doom in Hexen, switching to UDMF is easy: with your map open in UDB, just hit F2 and manually set a different format. While your map will still use some of the outdated non-UDMF methods for stuff, they'll still work because they're supported. After that you can choose to optimize parts of your map or just continue developing it.

Step 3: Choosing an archive format and editor

Doom resources (graphics, sprites, sounds, etc.), code and maps are kept inside archives. There are two **archive formats**: WAD and PK3. (PK3 is a ZIP file that had its extension manually changed to PK3.)

There are also two **types of archives**: maps and everything else. Maps can *only* be in WADs, while everything else can either be in WADs or PK3. “Everything else” means code, assets (sprites, sounds, even textures), and any text lumps (MAPINFO, GAMEINFO, LANGUAGE, ANIMDEFS, etc.).

Maps and map WADs are mostly covered in Step 2, so now we’ll take a look at everything else. If you’re making a gameplay mod that has no maps, or if you’re making a map that has custom monsters, sounds and sprites, you’ll have to deal with the “everything else” category, i.e. resources.

In order to put sprites, sounds and other resources into your archive, you’ll need to use **SLADE**—regardless of format.

If you’re just making a map with a few extra props and some new sounds, you can choose to not worry too much and just put those into your WAD. If you do that, you’ll have to respect lump order and put the resources between specific markers, such as S_START and S_END. A WAD doesn’t support subfolders, it’s an *ordered* archive, i.e. it uses a specific order to tell the game what’s where in it. That’s also why you can’t edit WADs with regular archive editors such as 7-zip.

However, if you’re making a gameplay mod without maps, or a project that has both maps *and* a lot of custom resources, ***using a WAD for resources is a very bad option.*** The reason is simple: as stated above, WADs don’t support subfolders and can’t be edited with a conventional archive editor, so they’re exceedingly confusing to navigate if you have a lot of resources. There are many other minor reasons, but this is the primary one, and trust me, it’s a big one.

Using PK3s is very easy: you just have to put your stuff into specific folders. I.e. sprites go into **sprites/**, textures go into **textures/**, shaders go into **shaders/**, and so on. It’s fully described on the wiki in this article: **Using ZIPs as WAD replacement**.

If you have a WAD with resources, you should move those resources into a PK3 as soon as possible—it’ll make your life much easier.

If you have a large mod that has both maps and gameplay changes, it’s a good idea to keep maps in a WAD file, and keep all resources in a separate PK3 file. That’s perfectly normal, plenty of mods consist of multiple files that have to be loaded alongside each other when played in GZDoom. It’s also possible to put each map into a separate .wad and place them into **maps/** folder: this is described in the article linked above. This method is somewhat awkward while you’re still developing your project, but once you’re ready to ship it, it may actually be a good idea.

Now, there are some changes that using a PK3 brings, and you have to be aware of them:

- If you're using simple 1:1 textures (i.e. the image file is used as a texture directly), with PK3 you do *NOT* have to define those textures in TEXTURES1 or such. Simply put your texture images under **textures/**, and you're done. However, you'll need to use the TEXTURES lump if you need to define composite textures (i.e. layered, textures made out of several images, etc.).
- PK3 is a ZIP. As such, first, you can't have files with identical names. Second, you don't have to delete extensions from your files, as opposed to WADs. Leave all those .png, .ogg, .txt alone. It also makes the archive more easily readable.
    - If you want to make multiple DECORATE or ZScript lumps, don't try to actually do that. Instead create a base decorate.txt or zscript.txt file, put the other files into a separate folder, such as **mymodname/**, and then use **#include "mymodname/pistol.txt"** in the base file to add all the code files you need. It's way more convenient.
- And one the greatest aspects of PK3 is that you can keep it unpacked! You don't have to put everything inside **mymod.pk3** and edit it with SLADE all the time. Instead you can unpack everything into a **mymod** *folder* and run that in GZDoom. This means you'll be able to edit all of your resources with regular editors, without the help of SLADE. You can edit your code in Notepad++, your images in Photoshop, etc., etc. You won't have to worry about SLADE possibly crashing or not saving your progress. This makes editing stuff, especially code, much faster and easier. You can pack your folder into a pk3 when you ready to ship your project.
    - GZDoom can load folders in the same way it loads WADs or PK3s or ZIPs: e.g. **gzdoom -iwad doom2.wad -file mymod** where **mymod** is the name of the *folder*.
    - Obviously, the folder will have to have exactly the same structure, as a PK3 would. It is literally just an unpacked PK3.
    - If you decide to use a dedicated text editor to edit zscript/decorate code, you'll need to use specific syntax highlighting. Mostly C or C# highlights work well, but for example you can download ZScript syntax highlighting for Notepad++ on the forums. The only downside here is that you won't have popup tooltips with information about functions.
    - SLADE *can* edit folders just like archives. Just use **File > Open Directory** and paste the path to the folder you want. Do note: if you're using SLADE to set image offsets or something like that, it's better to just open the folder with the images, not the whole mod's folder. First, it's faster. Second, it's safer: a crash won't potentially screw up everything.
    - REMEMBER that SLADE **does not create backups for folders**, only for archives. You can protect yourself from accidental changes by using GitHub.

It's worth noting that a lot of big (and not even that big) GZDoom projects nowadays have their GitHub repositories. If you're keeping your mod as an unpacked PK3-format folder, you can turn that folder into a GitHub repository, which will give you endless backups, easy version control and history of changes, convenient tools for releasing projects, working in

teams, and just making sure your work doesn't get lost to a hard drive failure. Seriously consider it. The basics of using GitHub are covered here.

Step 4: Gameplay scripting. DEHACKED, DECORATE, ZScript

DEHACKED is the oldest method of creating gameplay mods. It modifies some data used by Doom, so you can change monster and weapon behavior, and such. You cannot, however, add stuff in DEHACKED, only redefine existing objects and sprites. This doesn't really have many applications nowadays, unless you want a specific challenge of making a vanilla-compatible gameplay mod, such as Doom 4 Vanilla: a Doom 2016-styled mod compatible with vanilla Doom.

Moving on to the big guns: DECORATE and ZScript.

ZScript is an advanced scripting language used by GZDoom, sometimes described as a distant relative of C#, C++, Java and/or UnrealScript. It's an object-oriented language which uses classes to create anything in the game, from menus and HUD to weapons, monsters and other objects (actors). At the moment of writing ZScript is a *relatively* recent thing and it's the go-to language for any gameplay changes you might want to introduce in GZDoom.

DECORATE is an old and cut-down version of ZScript that can only work with *actors* (i.e. monsters, weapons and other objects spawned in maps). DECORATE is supported by GZDoom and Zandronum (although, Zandronum's version is much older and doesn't have a lot of functions compared to the GZDoom version).

If you're developing for GZDoom, using DECORATE is pointless. There's a common misconception that ZScript is supposedly harder to use than DECORATE. This is, in fact, not true: ZScript just has a much higher ceiling in terms of the available features and possibilities, but the skill floor is the same in both languages. ZScript also supports all of DECORATE features and functions, and uses very similar syntax (for example, compare the Zombieman code on ZDoom wiki in Decorate and ZScript to see the difference). In other words, if you know DECORATE, you're already ready to code ZScript.

More information about ZScript and why it's not that hard to use is provided in the **ZScript Basics guide:** https://github.com/jekyllgrim/ZScript_Basics. Apart from that, look at the articles listed here, look inside gzdoom.pk3 to find Doom/Hexen/Heretic/Strife classes as well as functions in ZScript, and later (probably much later) you'll want to take a look at this WIP ZScript documentation by marrub.