Making Half-Life Mods - Part 1

by Robin Walker

Originally appeared in VERC Collective in Feb 24, 2003 and archived in TWHL since August 1, 2008. Here is the text as it appears in TWHL. — *(this article is presented as written for and published at Stomped.com, March 15th, 2001) Robin Walker is the co-creator of the original Team Fortress mod for Quake. He was hired by Valve Software in 1998 and helped to design the company's official Half-Life mod Team Fortress Classic. He is now helping to design Valve's next announced game, Team Fortress 2.*

This article is designed to help you make a Half-Life MOD. First, it has some advice on starting a MOD, and assembling a team. Following that is a collection of helpful tips on figuring out the game design for your MOD. Finally, it has a step-by-step guide on how to tie up the loose ends and finish your MOD, which is actually the hardest part of MOD making.

STARTING OUT

Let's start by looking at how to assemble a team. The guiding rule here is to Keep it small. Managing a team of people is a full-time job, even when all the team resides in the same building. If you?re dealing with an on-line team, you can easily spend all your time managing it, and that means you won't be spending any time on making your MOD. Adding more people to the team doesn't mean more work will get done. The more people you have, the more time spent managing them. Team Fortress's core team was 3 people. Counter-Strike's core team is 1 person.

When looking for team members, try to only hire people you absolutely cannot survive without. Your first instinct might be to hire anyone who can code, or model, or make maps, and so on. But, for your first version, you probably won't need more than 1 person for each area of your MOD (code, sound, models, maps). You may not even need any new models, sounds, or maps. Don't hire anyone until you've seen examples of their work. Make sure they've actually finished things. If they're a modeler who's done 20 models, but they're all half-finished, you don't want them.

MOD DESIGN

As a MOD author, the most useful question you can ask yourself is «Why should someone play my MOD?» It's a hard question to answer truthfully, but if you can answer it well, you're on the right track. Think about what other MODs are out there, and what they offer. Does your MOD offer something new to the players? Is what you offer enough to entice players who are busy playing other MODs? Even if you can't answer this question, just thinking about it will probably help your MOD.

You have power commercial developers don't: You don't have to worry about the commercial viability of new gameplay styles. Commercial developers have to worry about appealing to retail,

breaking even, and other nasty things, which is why most games are slight modifications on already proven gameplay. But you don't. You can try out truly new gameplay ideas that just might become the Next Big Thing. This is your edge over commercial developers. Make your job easier by concentrating on this edge, and don't spend your time trying to compete in the areas that commercial products are strong in. Most MODs can't compete on a content level (maps, models, sounds, etc) with commercial products...they've got teams of artists with years of experience. Beat them with your gameplay. Players will play a MOD that has very little in the way of new content, but has really fun gameplay. Something many people don't realize is that Team Fortress 1 had almost no new art for a year after it was first released.

You have another power over commercial developers. **You can release much, much faster and more often than they can.** We've summarized this MOD development philosophy with the phrase, «Release soon, Release often.» Commercial developers work for 2-3 years, release their game, and hope to god people like it. You don't have to make that leap of faith. You can design your whole MOD, write 25% of it and polish it to a playable state, then release it and begin getting feedback immediately. Then you can start adding the rest of your design piece by piece, at the same time rolling in the player's feedback to the first version, and continue releasing every month or two. You're in touch with your players at all times, so you'll never be in the situation where you've spent a lot of time on something you're not sure your players will like. The trick is to cut your MOD up into slices. The initial version needs to be fun and playable, but doesn't need every cool feature you've thought of. Be careful. «Release soon» doesn't mean releasing bad quality stuff, it just means doing your MOD in small, polished increments. The first version of Counter-Strike didn't have half of the features they have now. The CS team released a high quality, but not big MOD. Over the past year, they've been regularly adding more and more features and, in response, their player base has just continued to grow and grow.

«Different is NOT always Better.» When thinking about your game design, don't fall into the trap of believing that «Different is Better.» There's no reason to rewrite the shotgun code and have a new shotgun model if it doesn't impact your game in any interesting way. Keep in mind the first question, «Why should someone play my MOD?» The answer, «My MOD has a new combat system, and a new movement system,» isn't necessarily a good answer. So your combat system is different that Half-Life's...OK? But is it better? Does it make your MOD more fun to play? Does a new movement system make the game more fun? Player's are used to existing systems, and making them learn another one needs to be worth it for them. So before you think about changing something, make sure you know you're changing it for the better, and that it'll make your MOD more fun. Don't be afraid to just leave something the same as it was in HL.

Create realistic goals for yourself. Think about how long it takes a commercial developer to make an FPS shooter with 10 weapons. If your MOD is going to have 40 weapons, you're making life really hard for yourself. The thing to keep in mind here is **«Quality over Quantity.»** Players would far prefer to have 10 unique, well balanced, and fun to use weapons than 40 unbalanced weapons, some of which are slightly tweaked versions of others.

Don't be afraid to cut content and features. If the MOD looks like it's never going to be finished, or there's some content that you don't think meets the quality of the rest of the MOD, then start cutting. During the development of HL at least 30% of the original features in the design were cut because it became obvious they were unattainable in our timeline, or because we decided they weren't worth their development time. As we said above, «Quality over Quantity.» Players would prefer having 3 really good, well play-tested maps over 10 untested ones, and it'll give your MOD a reputation for quality content. Don't let the world see your worst stuff.

Understand the engine. You really should read the documentation included in the SDK. The thing you'll learn most by doing so isn't whether you can do X with the engine, but rather how X should be done so it works well. You can make a gun that fires 50 rockets, but if you don't understand the way the engine works, you might do it in a way that significantly increases the network traffic your MOD uses. This is important for everyone in your MOD. If your mapmakers don't understand the engine, they'll make huge maps without any thought for how much network data will be sent to the players in them, and everyone will blame your code for being too network intensive. If you're a programmer, it's a good idea to join to HL Coders mailing list, where you'll be able to talk to many other MOD programmers, and a few Valve employees as well. The mailing list has archives going back a long way, which contain a lot of useful solutions to common MOD problems.

FINISHING

We see a lot of MODs that start out strong, produce a lot of great looking content, and never quite make the last step of getting it into the player's hands. This section will help you get into a release mode where you're driving towards producing a releasable version of your MOD.

We chose five weeks as a starting estimate of the time it'll take to get from normal development mode to a shippable version. It's likely you'll get better, and hence faster, at this with successive releases. If your MOD is larger in scope, or your team is substantially international, then it is likely to take more than five weeks, though the steps will be similar to the following.. If possible, try and get the team to commit a few hours of every day to the MOD for this period of time. If some team members can't do that, you're probably better off removing them from the shipping process. Get them to hand their part to someone else on the team who can put in the required effort. Shipping a product, even a small product, is hard and requires a substantial commitment.

There are many things in this section that might sound harsh or rigid. This is unfortunate, but a reflection on how hard a process this is. The advice here is a summation of lessons learned in the shipping of many products, and most of it a result of painful mistakes that set back our release dates. When you wonder whether a particular piece of advice here is necessary, it's possible that we once added weeks to our release date because we didn't take it.

This is also something that prospective employers are extremely interested in. It?s one thing to see that a MOD maker has produced a bunch of cool stuff, it's another thing entirely to see that they produced some cool stuff and actually shipped it out and people played it. The coolest map/model/code/sound/etc. in the world is useless if you couldn't go the last mile and ship it.

Fear not, this gets a lot easier once you've been through it a couple of times... by the third or fourth release of your MOD, you'll be an expert!

Continue to part 2>