Now that Schmovin’s archived, people will probably find this Google Doc somewhere in the repo.

This was written in June 2022.

Please take everything written here with a grain of salt.

UNFINISHED

Please come back when this meets the *definition* of “done”.

GAME PLAN

- Read some academic articles about sightreading, eye movement, or short-term memory
    - Cite ‘em

- Saccades (akin to “chunking”), smooth pursuit movements, vergence movements (inapplicable here), and vestibulo-ocular movements (may be inapplicable)
- Short-term memory vs working memory
-

Mods Design Notebook

Unofficial, “Peer” Reviewed, and Fact Unchecked

Applicable to NotITG modfiles and most other “funny” rhythm games

4mbr0s3 2, Spax (SpiritAxolotl), Ground Elmo (not to be confused with Round Elmo)

This document attempts to break down and analyze some design decisions, theory, and principles behind rhythm game “modcharts”.

A lot of these principles are easy to pick up on with enough experience with modcharts.

Note: This is not a tutorial on how to make modcharts.

What is a “*Modchart…*”?

Modchart is a term that tends to be thrown around rhythm game communities. As such, we have no *formal* definition for it.

Let’s try to find a good definition.

Urban Dictionary

“a rhythm game chart with effects that do such things like move the screen, make the receptors / strumline bounce, and various other things that make sight-reading the chart more difficult.” — yourtypicalataru

This is a pretty solid definition, giving some examples of certain effects, like the movement of rhythm game elements, and it highlights an ultimate purpose: to make the chart harder to read.

ITG Wiki

“Mod files (mods) are charts that apply effects to the chart to make it more challenging to read. Mod files can make the chart bounce around, move up and down, and more…”

This isn’t very different from the last definition, but it *is* from the ITG Wiki. Now, we know for sure that modcharts all have some element of “bouncing around” to make it harder for players to read.

GameBanana

“basically when the notes move and do funny stuffs”

…This is the best definition of a modchart.

Most people tend to generalize modcharts as being “bouncy”, but modcharts aren’t limited to that.

In fact, modcharts aren’t just limited to bringing visual challenges to players. They can bring new, fresh mechanics to what would otherwise be a normal rhythm game session. In extreme cases, they can even change the gameplay of a rhythm game entirely, adding new controls entirely!

With this lack of limitation, in what sense can we define a modchart? Can they be called a certain “game mode” of a rhythm game? Can they be considered new games entirely?

There *is* one common trait that almost all modcharts have in common: they introduce new things, and they only last within the gameplay session.

Once a modchart’s over, you won’t find any of those same fancy sight reading challenges when jumping into a regular chart in a game.

A modchart packs in a bunch of novel, challenging ideas into a neat package disguised as a single rhythm game chart.

Note the term “disguised”. The contrast between a modchart and a regular chart is *crucial*. A rhythm game sets a standard for its gameplay with its charts. Players will know what to expect every time they play a new song: a bunch of notes.

Your job is to break that standard. Catch them off guard. Bring that *Element of Surprise*.

One way modcharts bring surprise is by starting off normally:

- Most old ITG charts start off with the ITG2 background to blend in with other user-made charts. Some don't even introduce mods until later in the chart.
- Arcaea's April Fools charts start with a song transition and a normal-looking track, like all other charts.
-

Modcharts are NOT for the Photosensitive, Epileptic, Motion Sick, or the Faint of Heart

EXTREME PLAY MOTIONS ARE DANGEROUS.

Generally, if you play video games, it's kind of expected that you aren't any of the aforementioned.

Making things hard to see is generally the opposite of what anyone wants in terms of user interface design in a game, yet here we are: adding difficulty to a game not by means of making a chart harder by making it denser or more technical… but by making things harder to see.

Crazy…

But making things hard to see is not new… It's a *very* common game mechanic!

If you're reading this document to learn the art of making arrows harder to see, I hope you're ready to singe some eyes.

And if you're going to introduce NotITG to anyone, *make sure they can handle it.*

Introduction

by 4mbr0s3 2

Why am I making this document (with the help of some volunteers)?

Well, first of all, I've gotten into NotITG about a year ago, and I wanted a place to organize my thoughts about the design behind a lot of NotITG files. They're cool to look at, but how does one design a file to still be playable? In fact, this document was originally a kind of "personal wiki" (except it's a Google Doc) for taking notes, but I figured that it'd be more beneficial to actually release it.

Second of all, I wanted a document that can help less experienced people (who are interested in modcharts or may have been intrigued by some NotITG videos on YouTube) learn more about how people who play modcharts read them.

Third of all, I saw a few people make some Friday Night Funkin' mods (not to be confused with *note mods*) with modcharts. FNF mod templates (i.e. engines) like *Kade Engine* and *Psych Engine* came with some gameplay scripting features but didn't really mirror the way that ITG applies note mods.

"I have no idea what Psych Engine is"

Hence, I made Schmovin', a submodule that allows anyone to add ITG-like note mods to their own FNF mods. Schmovin' is already functionally complete, but I couldn't release it yet if people don't know basic mods design!

I made this document to address this.

Fourth of all, barely anyone else has made a proper guide to modcharts. There are millions of FNF modding tutorials but almost none that are for NotITG or mods design, so the latter is a bit inaccessible to people who are genuinely interested in making arrows move differently.

Game design isn't a straightforward art. It involves a lot of experimentation, and being aware of what one can work with (i.e. your tools) is a great way to start. In the following sections, I'll be covering what *I* think (in addition to what other contributors to this document think) makes modcharts readable and playable.

Hey! If you've read up to this point, you should join the NotITG Discord!

Know Your Audience

by 4mbr0s3 2

When making things, *please* keep your target audience for your things in mind. This applies to all things: Modfiles, essays, music, and *especially* games.

Easy modfiles are for beginners with not a lot of skill on modfiles. Hard modfiles are for experts with a lot of sight reading skills.

NotITG, being focused on modfiles, tends to have an audience that covers a wide range of skill levels, so it's safe to assume that a lot of them like more difficult modfiles.

Other rhythm game audiences, such as osu!, FNF, Clone Hero, or Arcaea, aren't used to modfiles, so tread cautiously and focus on making easier modcharts for those games.

An easy modfile can be boring for an expert. A hard modfile can be infuriating and even *sickening* to a beginner.

Imagine being reasonably good at a rhythm game and suddenly seeing someone else do 16th bursts…

Terminology

Let's just throw this in here for safe measure. Feel free to skip this section.

Dance/Rhythm Games

Dance Dance Revolution **(DDR)** — Series of music rhythm/dance games made by Konami.

StepMania **(SM)** — Open-source rhythm video game engine and clone of Dance Dance Revolution.

- In the Groove **(ITG)** — Series of music rhythm/dance games made by Roxor Games, Inc. based on StepMania 3.95. Got sued by Konami, canceling the third installment of the series. Introduced note modifier gimmicks with courses and Battle Mode.
    - **OpenITG (oITG)** — Open-source clone of ITG that attempts to replicate ITG's features as much as possible.
- **Not In The Groove (NotITG/nITG)** — Previously known as F***.exe, NotITG is a fork of OpenITG that adds more tools and features for making modfiles with Lua and XML. Check out the website!
    - **Ease Reader** — A general term for Lua modfile templates that read commands for easing specific note modifiers at different times of a song and puts it into the game (nITG/oITG). Can be thought of as a mods "timeline" script.
        - **Exschwasion Template** — Lua template that older modfiles use.
        - **Mirin Template** — A more recent, commonly used Lua template for modfiles made by XeroOl that allows for ease-of-use modfile creation with timeline functionality.

Friday Night Funkin' (FNF) — Open-source Kickstarter music rhythm game made by the Funkin' Team (ninjamuffin99, PhantomArcade, Kawai Sprite, evilsk8r). Originally made for game jam Ludum Dare 47. The game is now mostly associated with its modding scene as opposed to the original game.

- **Schmovin'** — Haxe library by 4mbr0s3 2 for FNF that implements modchart functionality into the game, including an ease reader, hold and note rendering, and some debug features. Its interface is loosely based on Mirin Template.
    - As of writing, it is NOT released (because it would be irresponsible to release it without informing people about mods/game design)

Pump It Up (PIU) — Korean music rhythm/dance games made by Andamiro. Like ITG, it is also based on StepMania, but uses five (5) panels and columns, wider timing windows, and different hold mechanics.

Gameplay Elements

Mine — A special ITG note that hurts your score and depletes your lifebar if its corresponding receptor is held. Can also be used to indicate specific tech patterns.

- **Shock Arrow** — DDR equivalent of an ITG mine.

Life meter — StepMania term for player health. Term derived from StepMania's source code. If it reaches zero, you fail the play.

- **Life bar** — Alternate term for life meter.
- **Life gauge** — Official DDR term for life meter.

Receptor — Official term for the stationary (sometimes flashing) arrows at the top (or bottom) of the screen for DDR, SM, and FNF that act as targets. Term derived from StepMania's source code.

- **Static note** — Alternate name for receptor. Term derived from FNF's source code.
- **Guide arrow** — Alternate name for receptor.
- **STEP ZONE** — Official DDR name for receptors.

Technicalities

Chart — Usually consists of notes placed on one-dimensional lanes played with buttons or panels.
Map — Usually consists of notes not placed on one-dimensional lanes (2D or 3D) and played with other apparatus (tablet, mouse, motion controls, etc.).

Stepfile — A StepMania file (.sm or .ssc) that consists of stepcharts usually of separate difficulties.
Stepchart — A StepMania chart.
Stepartist — An artist who makes stepfiles. Equivalent to "charter" or "chart designer".

NotITG / UKSRT

Modfile — A stepfile with note modifiers scripted with Lua. (Alias: Modchart)

UKSRT — UK Sight Reading Tournament, a series of gimmicky ITG tournaments from 2012 to present. There have been 10 tournaments since (not including spin offs), but 8 and above are more lore-focused. Check out the TV Tropes article.
WinDEU Hates You — Series of gimmicky ITG tournaments by WinDEU that usually followed normal ITG tournaments, featuring stories and lore that are an inexplicable mix of Friendship is Magic and Touhou Project.

Mawaru — Collection of Warioware-style minigame modfiles made by TaroNuke. See minigames.

NotITG Memes

Haunted House — See Haunted House.
SHAME Cube — Twisty cube gimmick introduced in UKSRT8's tang pong with a name that's also reminiscent of a funny meme GIF from RX - Bumble. A futuristic "2.0" version of the gimmick was introduced in FMS_Cat's modfile of Analys by Hayako (DJMAX RESPECT).
I Love Video Games (ILVG) — Refers to receptors in the order LUDR (as opposed to LDUR), which is a common arrow order for many DDR clones and can be thought of as "bootleg". Examples of games that use this order include Club Penguin's Dance Contest, Google's Champion Island Games, a few osu!mania arrow skins, etc. The term is a reference to an old Tee KO shirt by Exschwasion (?).
I Love SHAME — [redacted]

Ancient ITG

Black Lamp — Song by SD-501 from album *Maszkor Madness* charted for ITG pack *SPEEDCORE 2*. *Black lamp*.
Team Dragonforce — One of many old ITG teams, named after the band. Popularized by this video. TODO *More research needed*

WinDEX — 
Chris Foy — Former developer of ITG who charted many Expert songs. Now works at Step Revolution.
Kyle Ward — Former developer (sound designer) of ITG. Now producer of StepManiaX at Step Revolution.
VerTex² — Notoriously difficult ITG2 song featuring many stops and speed changes.

Sight Reading Charts and Patterns

The core gameplay of arrow games.

Reading Rhythms from Gaps

Let's get straight into the core gameplay before we mess around with gimmicks.

In all rhythm games, if you see a note approach and overlap its corresponding receptor or hit area, you press the corresponding button/panel/area.

In DDR/ITG, how do you know which panel to press if an arrow is approaching? There are two *key pieces of information* that the game tells you: the ***column*** the arrow is in and the ***shape/direction*** of the arrow!

Take away column information, and the player must rely on arrow shapes:

Take away arrow shapes, and the player must rely on columns:

(Keep in mind that spread players with non-arrow noteskins may not have much of a developed sense of which column corresponds to which direction or vice-versa, which can make sight reading for these players harder.)

Take away both, and it becomes too ambiguous.

How do you know when to press it? Often, people look at the size of the gap between the arrow and the receptor!

Most players can go far by just reading this distance, but you can't always trust it! What if the speed of the chart suddenly increases?

Or what if it suddenly stops?

Or what if the gap… did something else?

For rhythm games with column-colored or single color notes, this type of gimmick warrants some *speedy reaction times*. Players rely on seeing the gaps between notes and/or the receptors and judging when to hit them based on those gaps.

In DDR/ITG, however, the arrow ***colors*** are another key piece of information because they tell where an arrow is placed and when it needs to be pressed!

More on that in the next section…

Reading Rhythms from Color-Coded Arrows

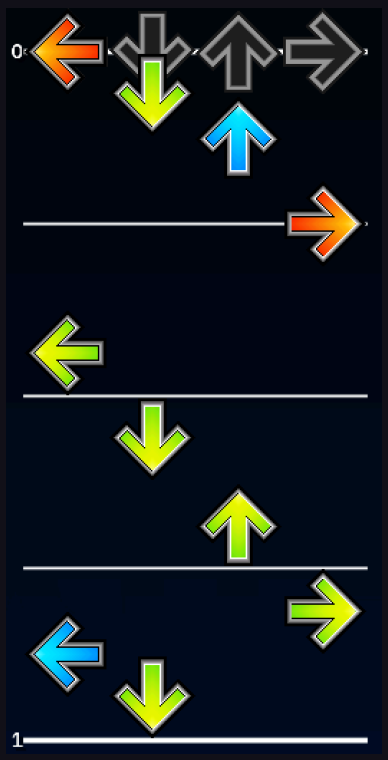
DDR/ITG arrows are colored based on where on the beat they are placed, which can also tell you when to hit on that beat.

[insert video]

Reds are 4th notes, which occur on an on-beat.
Blues are 8th notes, which occur on an off-beat.
Purples are 12th notes, aka triplets, and typically occur in songs with swing tempo.
Greens (ITG)/Yellows (DDR) are 16ths.
Oranges are 32nds
Teals are 64ths and higher

Exercise #2: Predict the rhythms (without even being familiar with them)!

Can you find out the rhythm here?



Answer:
[video of staircase pattern predictably being hit, shave and a haircut style]

That was easy, wasn't it?

This kind of gap sightreading can go a long way in DDR and ITG charts, but what happens if you can't rely on them?

Can you find out the rhythm here?
[video of staircase marching toward receptors]
Answer:
[video of staircase but with accelerate mod]

OK, that was a bit unfair.

What about here?
[video of staircase marching toward receptors]
Answer:
[video of staircase but Yukari Yakumo delays it by ½ beat]
…talk about sightreading *gaps*.

OK, last one.

Can you find out the rhythm here?
[video of staircase marching toward receptors]
Answer:
[video of staircase but arrows suddenly slow when approaching receptors]

Hah! Bet you didn't expect that, huh?
Even without mods, sightreading can still be made a challenge with *stops* and *scroll speed changes*.

So how do players read these rhythms? They look at the colors of the arrows and rely on the song's BPM.

DDR and SM (which includes ITG) all have arrows color-coded (or *quantized*) to their timing.

WIP

The Importance of Arrow Order

Left, down, up, and right. This is the exact order of the columns of arrows that began with *Dance Dance Revolution (DDR)*. Pure and unmodified. It's the *standard*. *StepMania (SM)*, *In the Groove (ITG)*, and *Friday Night Funkin' (FNF)* also use this order.

[Insert pic]

Although it may not seem like a big deal to spread players (who play with four buttons in a row corresponding to each column), this exact arrow order is what arrow keys players and pad players depend on when reading patterns because all patterns use this same order!

[insert pic contrasting a staircase pattern with another staircase pattern with RUDL]
Although the pattern shapes are the same, these are *very* different sequences of arrows!

The left arrow is on the very left, and the right arrow is on the very right. However, it seems arbitrary for the down arrow to always be on the left and the up arrow to be on the right. Some DDR clones mix up the up and down arrow positions as a result. More on that later…

Regardless, since DDR established this order, this is the exact order that most people are used to.

Reading Pad Patterns

DDR and ITG (and, consequently, OpenITG and NotITG) use a style of charting called pad charting: charts that are meant to be played on a *dance pad*.

Arrow keys players and pad players rely on recognizing different ITG/DDR patterns in these charts to be able to play them. Without seeing them, it'll be hard to play without good reaction time.

In fact, if you're not already familiar with pad charting, you should read this guide NOW!

ITG and DDR charts are more strictly defined with patterns than keyboard charts in games like *osu!mania*. As a result, with enough familiarity with ITG/DDR patterns, charts can be *predictable*.

In fact, Mawaru Simulator 2016 in UKSRT8 (see Minigames) has a minigame where you have to predict *missing notes* from patterns! In this case, staircase patterns.

Nue Houjuu in Mawaru Simulator 2016
Notice the missing step?
TODO: Better screenshot lol

Since ITG/DDR is played with a pad, ITG/DDR charts are also designed with the intention of having players *alternate* between their left and right feet. A lot of ITG/DDR patterns are defined by this alternation.

[insert pic]

Most arrow keys players emulate this pad play with both legs by playing with both of their index fingers. This is called the *index* playstyle, and it is the intended style for playing pad charts.

You can read a description of each keyboard playstyle here.

[insert pic]

Since ITG and DDR charts are mostly designed for leg alternation, positions that would cause awkward leg positions or body twisting are minimized… unless they aren't.

HyperTwist (ESP 15) from DDR A20 PLUS
I hope you like 270s!

Hence, patterns can't be completely random, and you *don't* have to memorize ITG patterns to be able to *predict* what arrow might be next in a sequence.

Exercise #1: Predict the patterns (without even being familiar with them)!

Try playing out each pattern with arrow keys with the aforementioned *index method*. Begin patterns that start with the left or right arrow with your corresponding index finger.

Chart Density

The lighter the density of a chart, the easier it is to read.
The denser the density of a chart, the harder it is to read.

Balance your charts with your mods, or vice-versa.

If you want to force players to read arrows differently than expected, such as with a D-Pad, or just have extreme mods applied, then having a lighter chart would balance out the difficulty.

On the other hand, if you have a heavy chart, it would make sense to balance it with less mods.

ITG Tech files, having dense charts, tend to use less mods.

NotITG files, having more mods, tend to use lighter charts.

Let's move away from arrows for a moment and take a look at how this Clone Hero modchart by Stargazer of Aegleseeker uses chart density.

Clone Hero uses disk-shaped notes (rather than arrows) and colors them based on column rather than timing. This means that players should be more reliant on the notes' column position and color and the gaps between notes for reading dense charts.

With having a mechanic where notes swerve across columns, the difficulty of reading the notes is balanced by the lightness of the chart. Players are given more time to process the *colors* and *paths* of the notes, and, since the notes come down at the *same speed* they would normally, the timing of the notes.

(Can you believe how much game design you can do with a video editor?)

More About Pattern Processing

Skills that relate to memorization and mentally processing, or deciphering, patterns of arrows in modchart gameplay.

Players Shouldn't Memorize Whole Charts!

First and foremost, players shouldn't have to memorize a *whole* chart to be able to play it! That's not very *fun*, and people are tired of having to explain that.
Only *stealth* players do that…

The Core of Sight Reading: Knowing Where to Look

What's the most important part about sight reading? Where you're looking, obviously!

Making sure that the player knows where to look is the most significant part about designing a modchart.

For the most part, beginners tend to look at notes closer to the receptors, and pros tend to look at notes farther in advance.

Most beginners take more time to process arrows and tend to process them as they come, while most pros take less time to process arrows and tend to process patterns a few beats farther. More on that in the next section…

“Chunking”

Most skilled players mentally process big *chunks* of patterns in advance to be able to play them without having to be distracted by random mods that may happen.

“Chunking” exercises *visual memory*.

In fact, you can practice “chunking”... right on Human Benchmark!
Instead of square patterns, however, you memorize chunks of ITG arrow patterns in NotITG. Since ITG patterns are also standard, you’ll find it much easier to memorize those.

Pattern Memorization

Modfiles can challenge players’ mental capacity by having them remember specific patterns or sequences of arrows.

In many files, a rhombus with a question mark (�) is often used to indicate specific patterns that need to be memorized.

The mod file of *Nyan Cat* by daniwellP by TaroNuke in Mods Boot Camp 3 demonstrates this very well…

Remeb[m]er this!

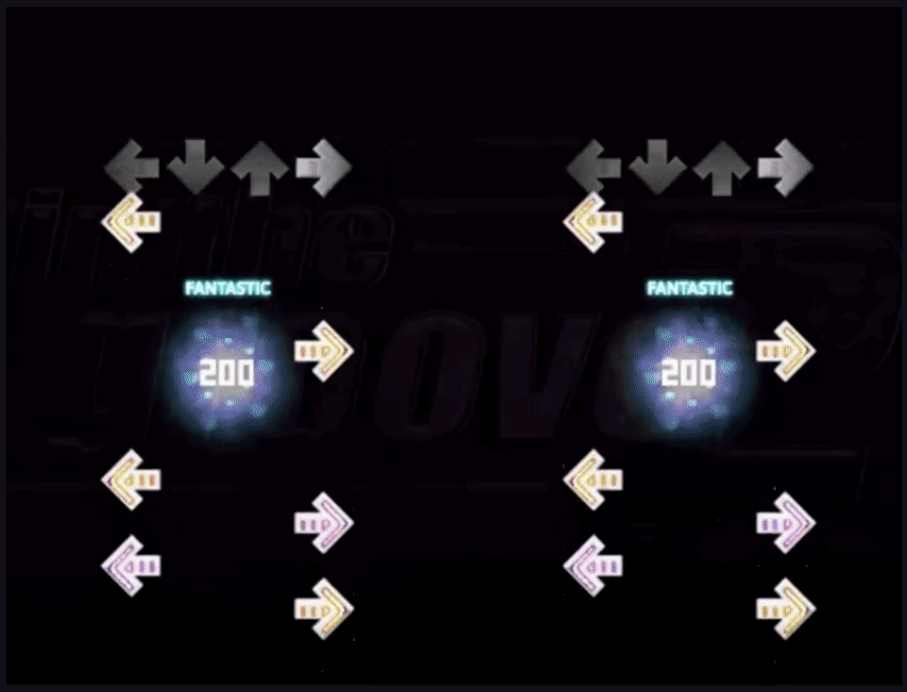
However, there are some cases where this idea of memorization can be taken… long-term. Even across tournaments!

Rhythm Memorization

Songs repeat. A lot. It's the basis of song structure.

You'll find that songs tend to use the same rhythms a lot, and you can take advantage of that with the way you design modcharts.

The final stage of UKSRT8, *get f**ked*, uses both rhythm and pattern memorization to test players' ability to maintain a beat in its absence.



UKSRT8
Going, going…

TaroNuke's modfile of Hypersurface by Grigori Poincaré (BilliumMoto) also demonstrates this by charting the up arrow to a constant rhythm in the song and making it optional.

Subconscious Memorization

How Exactly Do I Practice?

While there are some pro-tier players who are able to sightread and lifebar pass most official modfiles (see OISRT), it's not very realistic to expect regular players to lifebar pass challenging files without repeated practice.

Through playing a modfile multiple times, you learn what to expect and utilize…

- Muscle memory
- Pattern memory

Motions and Animation

How animating for modfiles can influence gameplay.

Different Types of Motion Stand Out

Motion Persistence

It's easier to time something that's in motion than by rhythm alone.
When an arrow disappears while moving, your brain can guess where it is or where it'll appear again.

Motion Timing

Something that's accelerating gives anticipation toward an earlier hit.

Rhythm Doctor by *7th Beat Games* takes advantage of this with SVT beats:

Bonus: Animate *to* a Beat, or *on* a Beat?

Animusic's website covers in detail what kinds of motion can result from or drive a sense of rhythm. I recommend reading it here.

Common Gimmicks

Gimmicks that you may often find in any arrow rhythm game.

Color Coding

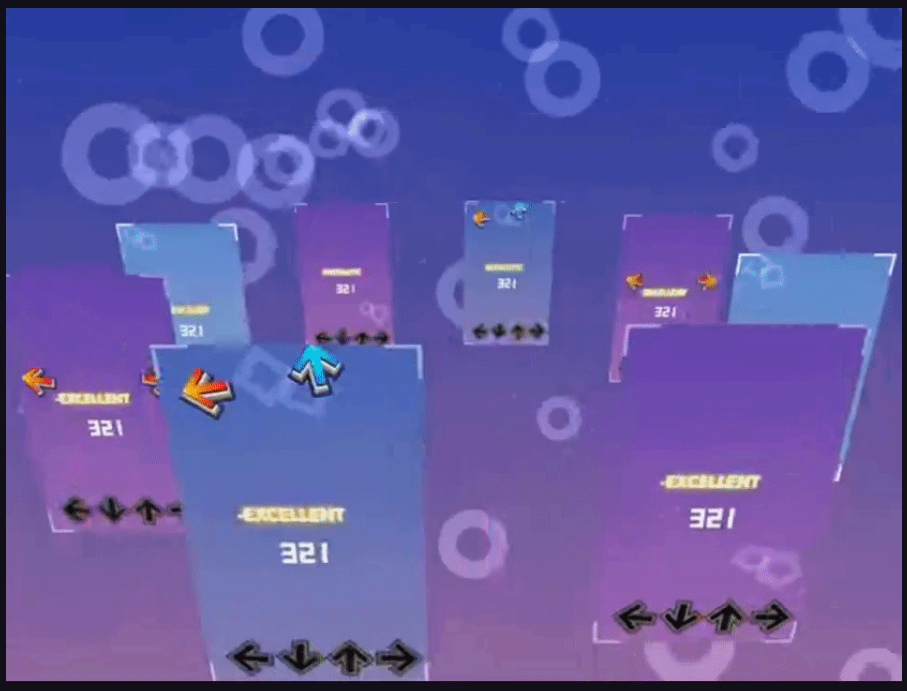
D-Pad

The d-pad gimmick resembles… a d-pad!
This gimmick is very common in NotITG files.
Sometimes it can rotate and do other funny things.

Spinning

“The world is spinning, spinning…”

Honestly, this is less of a “gimmick” than a common type of motion you’ll find. A lot of modfiles can involve wheel-like, spinning motions. You can have arrows or playfields spin on any kind of axis…



HALFBY - bubble attack (CHUNITHM STAR) modfile by Exschwasion (OISRT: Day 2)

NotITG Gimmicks

Gimmicks that you'll often find specifically in NotITG.

Spell Cards

Spell Cards are optional named sections that can be defined in a NotITG mod file that often denote when specific mods and effects are applied. A player's performance in these sections can be reviewed by pressing the up and down arrows on the *Evaluation* screen.

The name of the concept originates from the *Touhou Project* games, where spell cards are timed sections of bullet attack patterns (danmaku). In this way, the mods applied throughout a mod file can be seen as different "attacks" against players.

Haunted House

A Haunted House is a reaction-time gimmick that originates from Stage 2 of *WinDEU Hates You 401K* with the song *Haunted House* by cYsmix.

cYsmix - Haunted House modfile by WinDEU (WHY401K)

Here's the gimmick in its original context: (tournament footage / line-out)

Yep. You switch the arrows and the mines when the player least (or most?) expects it. Sometimes, you can tease them and psych them out.

It's a very common (almost overused) gimmick in NotITG files.

Although the gimmick is named after the song, the concept of "sudden mines" stemmed long before then.

Actor Proxies / Proxy Wall

In NotITG's engine, actor proxies are actors that copy the way that another actor is rendered. Player proxies copy player playfield actors and can be moved and duplicated anywhere.

Hence, modfile creators can create grids of said player proxies that can occupy the entire screen, known as *proxy walls*.

Proxy March

A variation of a proxy wall where several player proxies zoom past the camera, keeping players' eyes moving toward the farthest one, which has more screen time.

Masayoshi Iimori - Hurry Hurry

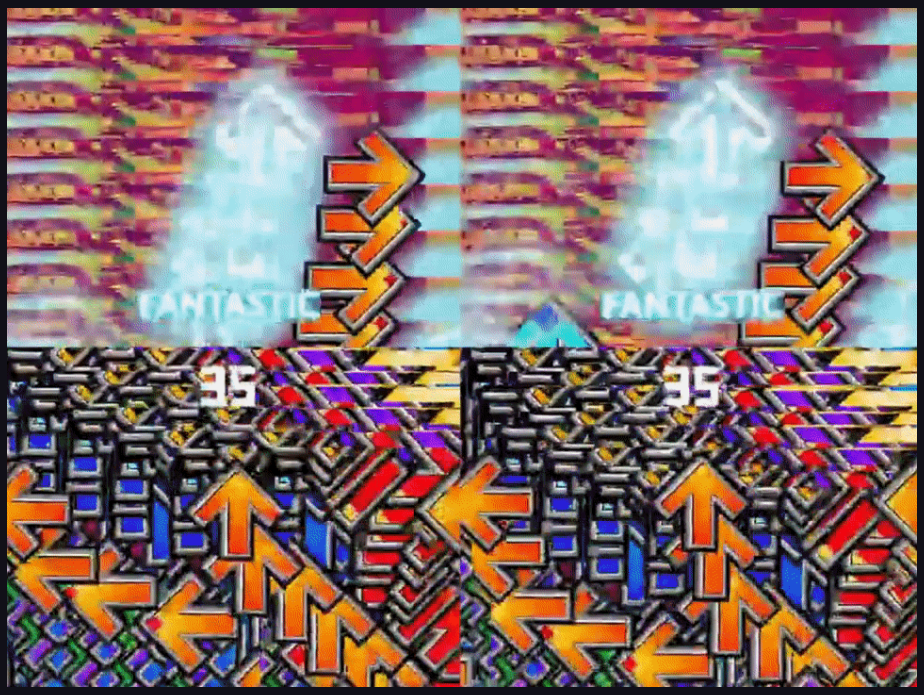
Actor Frame Textures / Framebuffer (Render Target) Shenanigans

A framebuffer in OpenGL is, basically, a place where things go to render.

Think of it like a camera. You can take a picture every frame and put it on screen, use that picture as a texture for something else that can be put on screen, or modify that picture in real-time before putting it on screen.
This is an oversimplification of what framebuffers are, but it's *good enough*.

Actor frame textures in NotITG are framebuffers.

One of the more common graphical effects that can be achieved with render targets is the Hall of Mirrors effect, where objects leave a trail against the background.



Frums - 19ZZ modfile by XeroOl & Sudospective (Mod Rush Couples)
This is an extreme example of the *Hall of Mirrors* effect.
(RIP bitrate)

Strobe / “Atari Pacman” Effect

Older versions of NotITG didn’t support more than two playfields. One old to simulate it is to flicker both playfields between two different positions to give the illusion that there are four.

One downside to this effect is that it’s dependent on framerate.

Strobing at higher fps is easier to read than at lower fps, which means that high refresh rate monitors will have an advantage.

Characters and Story

ITG sight reading tournaments often featured short plot lines.

Minigames

No hay pistas aquí!

How many kinds of games can you make up with only four digital inputs?

TaroNuke's *Mawaru Collection* is notorious for implementing various types of minigames in Warioware-esque mod files… all on StepMania's engine!

One Final Thing…

Mod files are meant to be played by other people (most of them, at least).

If you're not too sure your file is up to standards on what can be considered "playable" and/or "fun", you need to have other people *playtest* it!

Playtesting with the right audience guarantees useful feedback and information that will help improve your files. In fact, one very useful tool for this kind of information is an eye tracker.

Don't just have the *experts* playtest, though! Have people of all different skill levels playtest. If you're brave enough and you believe your file isn't very nauseating, have non-nITG players playtest too!

If people tell you to change your file in a certain way that *they* think would make it better, just remember that it's ultimately up to *you* to make mod design decisions.

More Questions?

Join the NotITG Discord!

Further Reading

Modfile Design

Beginner tips for improving charts!
Sudo's Guide to Mod Motion Design by Sudospective — Currently unfinished…?

Game Design

Think Like a Game Designer by Justin Gary — This four-hour read is great for getting into game design, especially because it isn't as expensive as a textbook!
Masahiro Sakurai on Creating Games — What better way to learn about games than from a legendary veteran game director? Sakurai focuses on the different dramatic design elements that bring a game to life.

SCRAPPED SECTION

Since this document isn't focused mainly on StepMania, it's probably best scrapping this.

~~Bonus: History of the StepMania Modding Scene~~

~~It will be a *really ambitious undertaking* to document the timeline of events that led to the creation of NotITG but it will *definitely* help people understand the historical context of sight reading tournaments.~~

~~If this gets too big, this might need to be a separate document.~~

~~TODO:~~
~~Research~~
- ~~ITG .crs files~~
- ~~BrotherMojo~~
- ~~AnimeLeague~~
- ~~Team Proof of Concept "Era"~~
- ~~—~~

~~Primary Sources~~
- ~~UKSRT Discord Server (Check History)~~
- ~~Honestly the devs themselves~~

~~Secondary Sources~~
- ~~Descriptions of ten-year-old ITG videos~~

Bonus: List of *Mod* Sight Reading Tournaments & Lore (for Reference)

Format: Month Year — Tournament Name — Subtitle

UK Sightreading Tournament

June 2012 — UKSRT —
June 2012 — UKSRT2 — Stepman Paradise
June 2012 — UKSRT3 — You Can (not) Hit Arrows
June 2012 — UKSRT4
June 2012 — UKSRT5
June 2012 — UKSRT6
June 2012 — UKSRT6.5
June 2012 — UKSRT7
June 2012 — UKSRT Summer 2015
June 2012 — UKSRTALE
June 2012 — UKSRT8
June 2012 — Hardware Bullshit Tournament (UKSRT8.1)
June 2012 — UKSRT9
June 2012 — UKSRTX

Modster Mash

June 2012 — Modster Mash
June 2012 — Modster Mash 2
June 2012 — Modster Mash 3 — Galactic Rumble

Mods Boot Camp

June 2012 — Mods Boot Camp 3

WinDEU Hates You

~~Other SRTs~~

~~June 2021 — OISRT~~