GTC – Ginren Traffic Control
簡易ドキュメント　for V0.4β

ドキュメント目次

0: はじめに

この度は、拙作 Mod「GTC - Ginren traffic control」をダウンロードいただき、ありがとうございます。当 Mod を使用するにあたっての方法・ならびに注意事項をまとめております。一度必ず目を通していただきますようお願いいたします。

0-1: 当ドキュメント内の記述において

当ドキュメントでは、以下のような名称を使用することがあります。

- 「開発者」とは、当 Mod のメイン開発者である「銀河連邦」を指します。
- 「銀連製作所」とは、「株式会社 銀連製作所」のことを指します。
- 「利用者」とは、当 Mod を使用して Minecraft を楽しむ方、当 Mod を改造する方、当 Mod の拡張セットを作成する方を総称した呼称です。
- 「Minecraft」「MC」とは、特別に明記しない限り「Minecraft Java Edition」を指します。

0-2: 当ドキュメントの更新について

Mod の開発とは分割して作業しているため、Mod の更新に合わせて更新されることもあれば、全く関係ない時期に更新されたりすることもあります。定期的にご確認することをお勧めいたします。最新版は常に配布サイトにて掲載されています。このドキュメントが製作された時点での配布先は

https://ginren.info/G-factory/Addon/GTC.php

となります。URL は変更される可能性もあり、また予告なく閉鎖される可能性もあるためご了承ください。

0-3: This document is JP-only.

For English, this document is not provide English's document because a developer can't use English well. If you can't read this document, I'm sorry but please translate this. And, the developer do his best to understand English, but he can't answer all of English questions by E-mail, or Twitter.

このドキュメントは日本語専用です。英語圏にお住まいの方はお手数ですが Google 翻訳などを使用し翻訳してください。開発者もできるだけ英語でのお問い合わせを理解しようと頑張りますが、それでも全部の問い合わせに回答できるとは限りません。あらかじめご了承ください。

1: このModについて

当 Mod「GTC - Ginren traffic control」は、ngt5479 氏が開発・公開されている「Real Train Mod」、通称「RTM」で追加される信号機との連携を主とした、交通機器を多数追加する Mod となっています。当初は制御機だけのつもりでしたが、交通機器の追加でより Minecraft の世界をリアル化しようという目的にシフトしつつあり、Mod 名の意味が薄れてきてしまっています…。

今までレッドストーンを使用して広大な土地、もしくは地下に広大な回路を頑張って作り、やっとこさ信号機を動作させてきたと思います。そんな無駄遣いからもおさらば。この Mod を使用すれば、1 ブロックの制御機にコネクタを差し込むだけで信号機が動作します。

ただし、仕様制約上「2 種類の切り替わり」しか表現できません。「3 種類以上の切り替わり」を表現するためには、従来通り回路でどうにかする必要があります。ようは、「主道・従道・歩灯」の切り替わりにしか対応していないということです。今後は追加も考えていますが、なかなか厳しいところです。

このように、RTM で追加された信号機を余計な回路なしで操れるようになります。Minecraft の世界がより現実に近くなると思われます。

余談ですが、当 Mod は当初公開する予定はなく、開発者の街づくりのためだけに製作していたものでした。

1-1: 初めてでもすぐに使えるプリセット

　押ボタン式信号機、定周期式信号機をあらかじめプリセット回路として記憶させています。制御機を設置したら、必要なパラメータ（秒数とか）を指定するだけで勝手に動作を開始します。凝ったことをしたい場合は、次に示すカスタムスクリプトを使用できます。

1-2: カスタムスクリプトで動作は無限大

カスタムスクリプトとして、JavaScript を記述することで、信号機の動作を定義させることができます。**Minecraft の動作中にでも変えられる**仕様となっており、**いちいち再起動しなくてもスクリプトを制御機から指定してあげるだけで即座に制御を変更することができる**ようになっています。カスタムスクリプトを作る方向けのマニュアルも当ドキュメントに含まれていますので、余裕があればチャレンジしてみてください。

なお、配布サイトにて開発者が製作したカスタムスクリプトも一部公開しております。導入方法については後述します。

また、カスタムスクリプトは JavaScript が分からない方でも作れるように、ツールなども公開したいと考えています（これに関しては需要あるのかわからないのでまだ未定ですが…）。

1-3: 制御機のほかにも機器類が追加

Ver0.2β より、「押ボタン箱」が追加されています。押ボタン箱は、押すときちんと「ピリリリリリ…」と電子音が鳴り、近くの制御機が動作を開始します。

レンダリングの影響で浮かぶのはもはや仕様ですが、GRP（拙作モデルパック）での技術を活かし、なるべく現実に似せたものに仕上げてあります。Mod 開発経験のある方は、モデルを差し替えることによってオリジナルの押ボタン箱にすることも可能です。

まだまだ機能はたくさんありますが、詳しくはご自身の目でお確かめください。

2: 導入方法について

GTC を導入する方法は、非常に簡単です。ただし、**シングルプレイを前提としたつくりにしているため、サーバーでの動作は保証できません**。あらかじめご了承ください。開発環境ではサーバーテストは行っておりません。以下は、シングルプレイでの導入方法となります。

スクリーンショットは Windows10 のものを載せております。OS 固有の設定などはないのでだいたい同じ手順で入れれば動きます。

2-0: RTM･NGTLib をダウンロードする

当 Mod は RTM に依存しているため、先に RTM をダウンロードする必要があります。また、RTM の前提 Mod として NGTLib が要求されているので、そちらもダウンロードする必要があります。導入方法に関しては、作者様の配布ページをご覧ください。ここでは割愛します。

2-1: GTC をダウンロードする

配布サイトにアクセスしてください（上記に記されています）。配布サイトから、最新バージョンの GTC をダウンロードします。中身は「jar ファイル」になっています。

このような場所から、「GTC 0.4β（バージョンは最新バージョンで）」をダウンロードしましょう。「GingaCore」に関しては、前提 Mod として必要になるのでダウンロードしておくことを強く推奨します（しなくても、後述する手順を踏めばダウンロードできます）。

2-2: mods フォルダに入れる

ダウンロードできたでしょうか？　ダウンロードができましたら、その「jar ファイル」を動かしたい Minecraft の「mods」フォルダに入れてください。ようは RTM と同じところに入れておけば OK です。GingaCore をダウンロードされた方は、それも一緒に mods フォルダに入れてください。

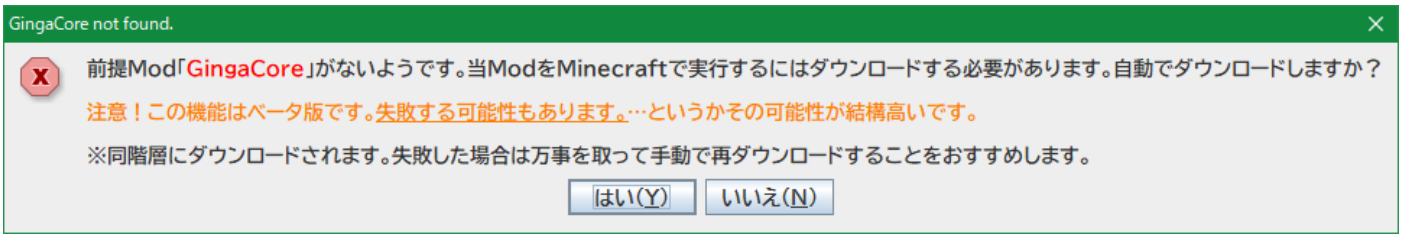
こんな感じで入れます。

2-3: GTC をダブルクリックする

この操作は任意ですが、GingaCore をダウンロードしていない場合は必ずしなくてはなりません。ダウンロードしている場合でも一度やっておくことをお勧めします。

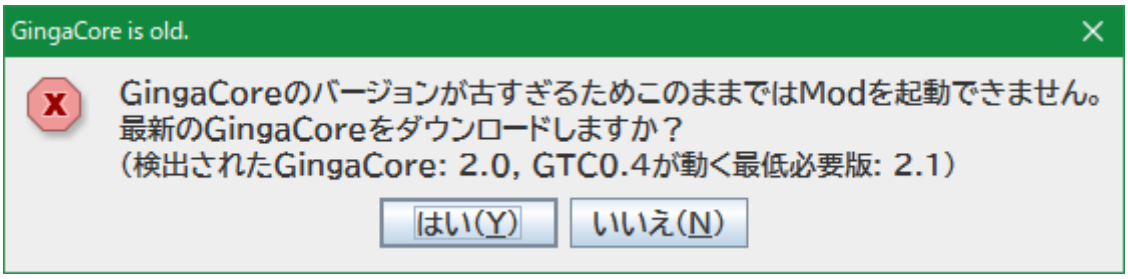
mods フォルダに入れた GTC をダブルクリックして起動してください。Java が入っていれば立ち上がります。起動時、自動でこの Mod の整合性を確認してくれます。

もし、GingaCore が mods フォルダに入っていない場合は、以下のような表示になります。



「はい」をクリックすると、自動で mods フォルダにその GTC のバージョンに適合する GingaCore がダウンロードされます。ただ、ダイアログにも書いてある通りこの機能はベータ版です。通信環境によっては失敗することもあります。一応開発環境では失敗はほとんどありませんでしたが、失敗した場合は手動でダウンロードください。

また、GingaCore が導入されていてもその GingaCore が古い場合は以下のような表示になります。



「はい」をクリックすると、mods フォルダに最新の GingaCore がダウンロードされ、既存の（古い）GingaCore は削除されます。この機能もベータ版なので消えなかったりしますが、その際は手動で削除してください。メッセージには最低必要版としてバージョン名が書かれています。

Ver0.4β 現在、GTC 自体の更新をチェックするスクリプトは導入していません。いずれ追加します。

2-4: (任意)スクリプトフォルダを作成する

これはあくまで任意で、しなくてもいいんですが、カスタムスクリプトを使用するときの為に「mods」フォルダの中に何でもいいのでわかる名前を付けておきましょう。「GTC」とかその辺でいいと思います。のちに使います。

2-5: Minecraft を起動してみる

これで導入は完了しています。実際に Minecraft を起動してみましょう。無事にタイトル画面まで進むはずです。mcmod.info を作成していないため、現時点では以下のように Mod 詳細を開いても空っぽですが、GingaCore と GTC が両方認識されていればひとまず導入成功です。なお、エラーが出た場合は後述するトラブルシューティングを参照してください。

3: 制御機の使い方

このModのメイン機能である、制御機の使い方を紹介します。

初期状態はこのように真っ白の謎の物体のごとく制御機が登場します。この状態ではまだ動きません（正確に言うと超絶怒涛の速さで定周期制御を行っています）。ちなみに制御機はオリジナルのタブに入っています。

3-1: GUIの見方

設置された制御機を右クリックすると、以下のようなGUIが開きます。

制御機の設定を行えるGUIとなっています。（開発環境は英語版で行っていますが日本語は一応ほとんど翻訳しているはずです）

将来的に若干GUIの変更はあるかと思いますが、概ねこのようなGUIで落ち着くでしょう~~（GUI構築しなおすのめんどくさい）~~。

まあ細かいところは読めばわかると思うんですが、次ページにてパラメータの大まかな内容を紹介します。

3-1-1: 青時間(Green time)

その名の通り、**主道側**が青を点灯し続ける時間となります。単位は秒で指定します。

3-1-2: 黄色時間(Yellow Time)

こちらは主道・従道共通で、黄色の点灯時間を秒で指定します。現実世界での参考値としては、概ね 3 秒～4 秒が標準となっています。

3-1-3: 赤時間(Red time)

これは少々ネックで、**主道**が赤を点灯し続ける時間を秒で指定するんですが、側道の時間はこの赤時間に合わせて自動計算されるため、側道の長さを考慮したうえで考える必要があります。側道と主道の関係に関しては後述します。

3-1-4: 全赤時間(All Red Time)

全方向が赤となっている時間を秒で指定します。現実世界では見切り発進を防ぐために導入されています。幅の広い交差点では 3 秒～5 秒、通常の交差点では 2 秒であることがほとんどです。

3-1-5: 点滅時間(Flush time)

歩灯が点滅する時間を秒で指定します。定周期式のみの制御機を使用している方はこの値は関係ありません。

3-1-6: カスタム数値(Custom)

カスタムスクリプトで指示がある場合に使用する数値を入力します。通常は 0 を指定します。現在は数値のみ対応しています。複数入力することができます。複数入力する場合は、「,」で区切って入力します。**スペースは入力してはいけません**（入力できないようにしていますが）。

例えば「1 と 2 と 4」を指定したい場合は、「1,2,4」と指定します。順番が大事なので正しく入力してください。なお組み込みの制御ではこの値は使用しません。

3-1-7: 警察署番号(Police)

お遊び機能として、神奈川県方式の管理番号をつけることができます。その際に指定する警察署番号となります。なお、この管理番号はもしかしたら系統制御などの実装で使われるかもしれないので、可能なら何かしらつけておくことをお勧めします。被った場合の動作は保証できません。

3-1-8: 交差点番号(Number)

これも警察署番号と同じく管理番号の指定方法です。交差点番号と警察署番号を合わせてユニークな番号を生成します。両方入力しないと印字はおかしくなります。

3-1-9: カスタムスクリプトパス（Path）

制御機タイプを「CUSTOM」に設定すると出現する GUI です。パスには、有効なカスタムスクリプトのパスを入力します。パスの指定については後述します。

3-1-10: 点滅動作の設定

チェックボックスが 2 つあると思います。それぞれ、チェックをすることで「昼間点滅」「夜間点滅」の動作にすることができます。どっちもチェックすると常時点滅になります。宮城県の UFO 信号機の最期みたいな制御ですね（**マニアにしかわからないですね**）。

3-1-11: 制御機のタイプ（Type）

この右下にあるボタンをクリックするたびに、プリセットの制御機タイプを入れ替えることができます。現時点で対応しているのは、以下の通りとなっています。

- CONSTANT（定周期、車灯のみ）。歩灯がない場合はこっちの方がやや軽いです
- CONSTANT_PV（定周期、歩灯あり）。
- DETECTED_PV（押ボタン式信号機）。
- ~~DETECTED_CAR（車両感知式）~~。Ver0.5β で廃止予定です。

もう 1 回押すと「CUSTOM」となり、カスタムスクリプトを使用することができるようになります。さらにもう 1 回押すと定周期に戻ります。

3-1-12: 各パラメーター共通の注意点について

コピペツールなどで**日本語とか数字以外の文字を入力すると落ちます**。また、すべてのパラメーターにおいて 0 を入力した際の動作は保証しません。負の数はもってのほかです。

また、計算上側道の時間が負になったりした場合の動作も未定です。一生変わらなかったりするのでお気を付けください。

3-2: カスタムスクリプトの指定について

上記「CUSTOM」の欄で指定するパスについて。パスは基本的に「mods」フォルダ内のスクリプトを参照します。**「mods」フォルダを基点として相対パスで指定することを推奨しています**。ただし、それ以外の場所においても認識する場合があります。捜索方法は以下の順番となります（が、この順番は変更されたりするため必ずこうなるとは限りません！　この順番に依存しないようにしてください）。

1. mods フォルダを基点とした相対パスとみなします。
2. 見つからない場合、Minecraft の実行フォルダを基点として検索します。
3. それでも見つからない場合、この Mod が読み込んでいるビルトインリソースから読み込もうとします。
4. リソースが存在しない場合は、最終手段としてシステムドライブを検索します。
5. 1~4 を行っても見つからない場合は、OK を押しても画面を閉じる（保存する）ことができません（出力ログを見るとエラーが出ていることが分かります）。

例えば、「C:\Users\Steve\Minecraft」で Minecraft が起動しているとします。その際に、CUSTOM で「custom.js」を指定したとします。その際、以下のように捜索が行われます。

1. 「C:\Users\Steve\Minecraft\mods\custom.js」を検索します。
2. 存在しない場合、「C:\Users\Steve\Minecraft\custom.js」を検索します。
3. まだ存在しない場合、「C:\Users\Steve\Minecraft\mods\GTC-X.X.jar 内のリソースを検索します。
4. それでも存在しない場合、最終手段として「C:\custom.js」を検索します。
5. それでも見つからない場合はこれ以上検索しません。

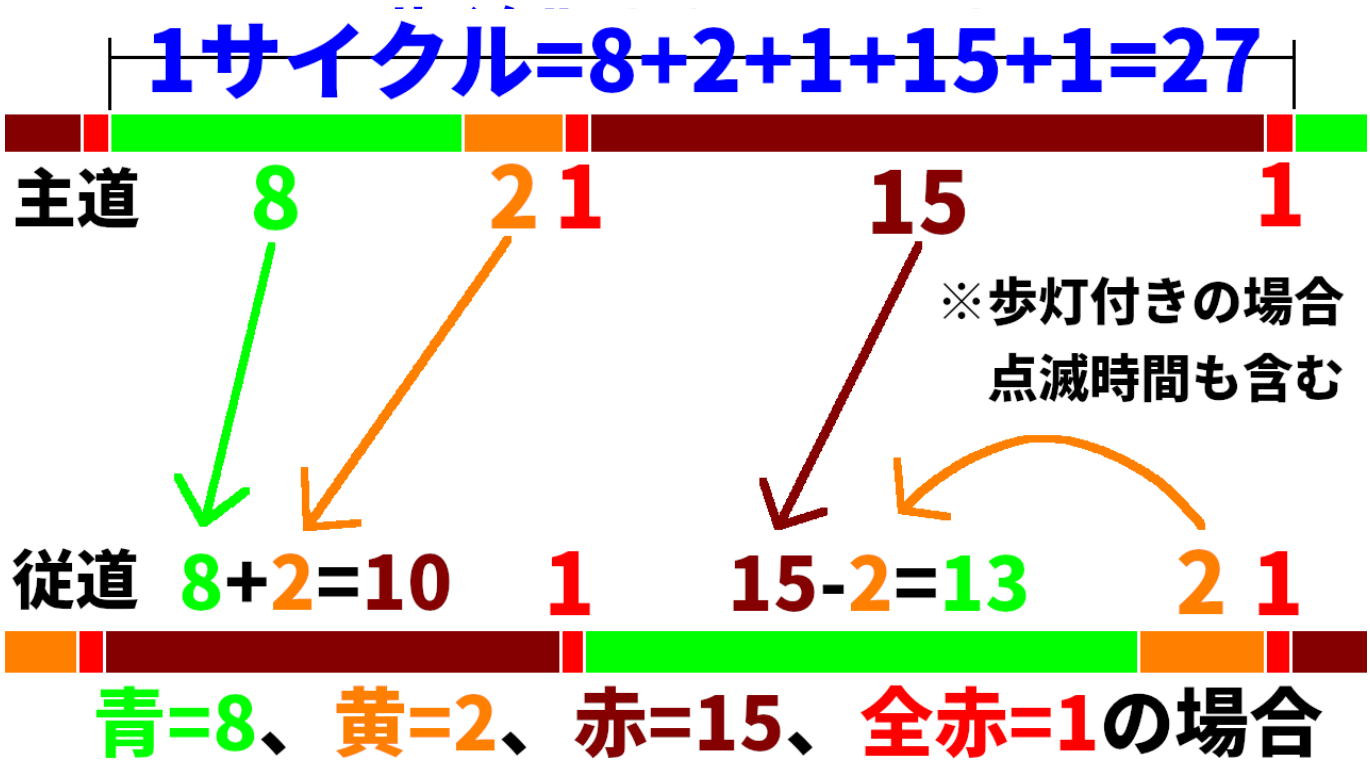
ちなみに、全然別の場所にパスを指定することもできます。その際は、「絶対パスで」指定します。絶対パスとは、Windows なら「C:\」とか「D:\」とかで始まるパスです。

絶対パスとか相対パスとかよくわかんないって方は、以下のようにスクリプトを置いて使用しましょう。

1. mods フォルダの中に「GTC」というフォルダを作る（導入のところでやってますね）
2. その中にスクリプトを入れる（例として、custom.js を入れたものとします）
3. CUSTOM に「GTC/custom.js」と記述する

なお、見つからない場合は出力ログ（ランチャーの設定で表示できます）にエラーが表示されますのでご確認ください。また、**制御機が稼働している間はずっとそのファイルを移動させてはいけません**。次回読み込み時にその制御機スクリプトが存在しない場合は硬直してしまいます。

3-3: 主道と側道の時間について



この画像がすべてを表しています。<u>すべては「主道」の時間を基準として行われます</u>。また、<u>主道の時間しか指定することができません</u>。これを見てもわかんない方向けに、計算式を書いておきました。参考にしてください。

3-3-1: 定周期式制御機(歩灯無し)の場合の計算式

青時間→お好きな時間を指定してください

黄色時間→お好きな時間を指定してください

赤時間→側道を青にしたい時間＋黄色時間

全赤時間→お好きな時間を指定してください

定周期制御機の場合は特にややこしいことはないのでこんな感じになります。1 サイクルの時間は、**青時間＋黄色時間×2＋赤時間＋全赤時間×2** となります。

以下、例です。

青時間→30 秒

黄色時間→3 秒

赤時間→側道 20 秒青＋黄色時間 3 秒＝23 秒

全赤時間→2 秒

サイクルの合計時間＝30+3x2+23+2x2=30+23+10=63 秒

3-3-2: 定周期式制御機(歩灯あり)の場合の計算式

歩灯が入ってくるため若干めんどくさくなります。

青時間→主道車灯青時間＋歩灯点滅時間

黄色時間→お好きな時間を指定してください

赤時間→側道車灯青時間＋歩灯点滅時間＋黄色時間＋全赤時間

全赤時間→お好きな時間を指定してください

点滅時間→お好きな時間を指定してください

歩灯の点滅が入ることにより、赤時間の計算がややこしいことになっています。1 サイクル当たりの合計時間は上記と同じで、

青時間＋黄色時間 x2＋赤時間＋全赤時間 x2

となります。例えば、車灯を 10 秒青、点滅時間 5 秒、黄色時間 3 秒、全赤時間 2 秒、側道の青時間を 15 秒とします。単位はめんどくさいので全部秒を入れてみてください。

青時間→10+5=**15**

黄色時間→**3**

赤時間→15+5+**3**+2=25

全赤時間→2

点滅時間→5

1 サイクル当たり、15+3x2+25+2x2=15+25+10=45

定周期式はこのような感じの計算式となります。

3-3-3: 押ボタン式制御の場合の計算式

押ボタン式の場合、すこしパラメーターの見方が異なります。まず考慮すべき長さを書き出してみます。

- 歩灯が青である時間
- 歩灯が点滅する時間
- 車灯が黄色である時間
- 全赤時間
- サイクルが終了してから連続で切り替わる最低待ち時間

車灯は常時青なので、青時間を考慮する必要がありません。代わりに、一番下に記してある**待ち時間**が必要となります。当 Mod の押ボタン式制御（組み込み）は、押ボタンを押すと**即座に**サイクルが始まるように設計されています。したがって、待ち時間がもし 0 だった場合、歩灯が赤になりやっと車灯が青になったと思ったら歩行者にボタンを押されてすぐに赤になってしまうといった現象が発生します。それを避けるため、この待ち時間の間は**絶対に青であることが保証される**時間を指定する必要があるのです。先述の通り、この制御では青時間自体を必要としないため、**青時間の値を待ち時間の値として使用します**。

これを踏まえて、押ボタン式の際の設定計算式を見てみましょう。

青時間→最低待ち時間（この時間までは絶対に青が保証されます）

黄色時間→お好きな時間を指定してください

赤時間→歩灯青時間＋歩灯点滅時間

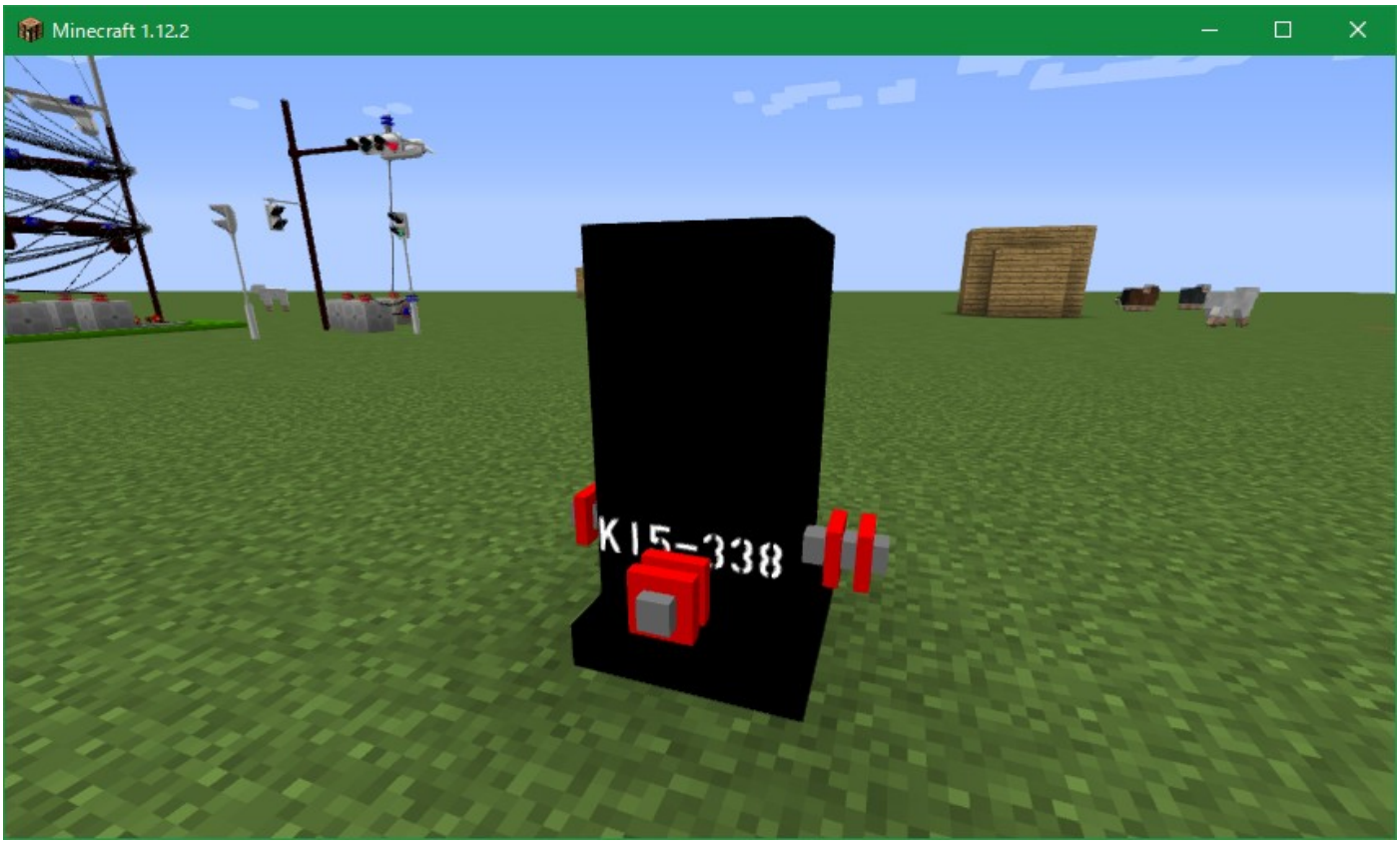
全赤時間→お好きな時間を指定してください

点滅時間→お好きな時間を指定してください

計算式自体は非常にシンプルです。例えるまでもありませんが、一応記しておくと、青時間を 10 秒にした場合は、歩灯が赤になり、車灯が青になってから 10 秒間はボタンを押しても切り替わらない（10 秒後にボタンが押されていれば切り替わる）ことになります。よくわかんなかったら青時間は 5 秒くらいにしておくといいです（現実世界ではありえませんが）。

3-4: 制御機と信号機をリンクする

ここまでできたらもう制御機自体は動き始めています。ただ、信号機とケーブルで結んであげないと、信号を伝えることができません。なので、制御機に**RTM のコネクタをぶっ刺す**んですが…。そのまま単純にデフォルトのコネクタを差し込むとこうなります。



ゲテモノ大感謝祭です。

こうならないためにも、RTM に拙作モデルパック「GRP - Ginren RTM Pack」を導入して、「見えないコネクター」で接続することをお勧めします。こんな制御機気持ち悪すぎます。

コネクタなしに動作させることができないので、仕方ないんです。ステマとかじゃなくて、真面目にＧＲＰ入れてください。お願いします。

さて、制御機には向きがあります。必ず設置した際に同じ面が向くはずです。これには訳がありまして、制御機の面によって４つの信号に分かれているからです。その一覧を紹介します。

まず、制御機の印字を表示したほうが分かりやすいので、何でもいいので番号をつけましょう。すると、印字が見えるはずです。その際に、以下の 2 つの印字が見えるかと思われます（ここで、印字は K12-345 と仮定します）。

K12-345
K12-345

この中で、上の方のフォントで印字されている面を正面とします（ステンシルみたいになっているやつ）。この面が、制御機を設置したときに自分の方向を向いています。これを基準に、上と下を除いた 4 面にコネクタを差し込むことができます（一応上にも差せるんですが意味がありません）。で、その 4 面の対応表は以下のようになっています。適切な信号機にコードをつなげてください。

- 正面→主道の車灯
- 背面→主道の歩灯
- 右側→側道の車灯
- 左側→側道の歩灯

ちなみに、これをあえて付け替えることによってサイクルを入れ替えることもできます。方角での指定ではないのでご注意ください。繰り返しますが、正面はステンシルフォントの印字がされている面です。

もうこれでもわかんなかったら適当につければ 1/256 の確率でヒットするので試してみてください。

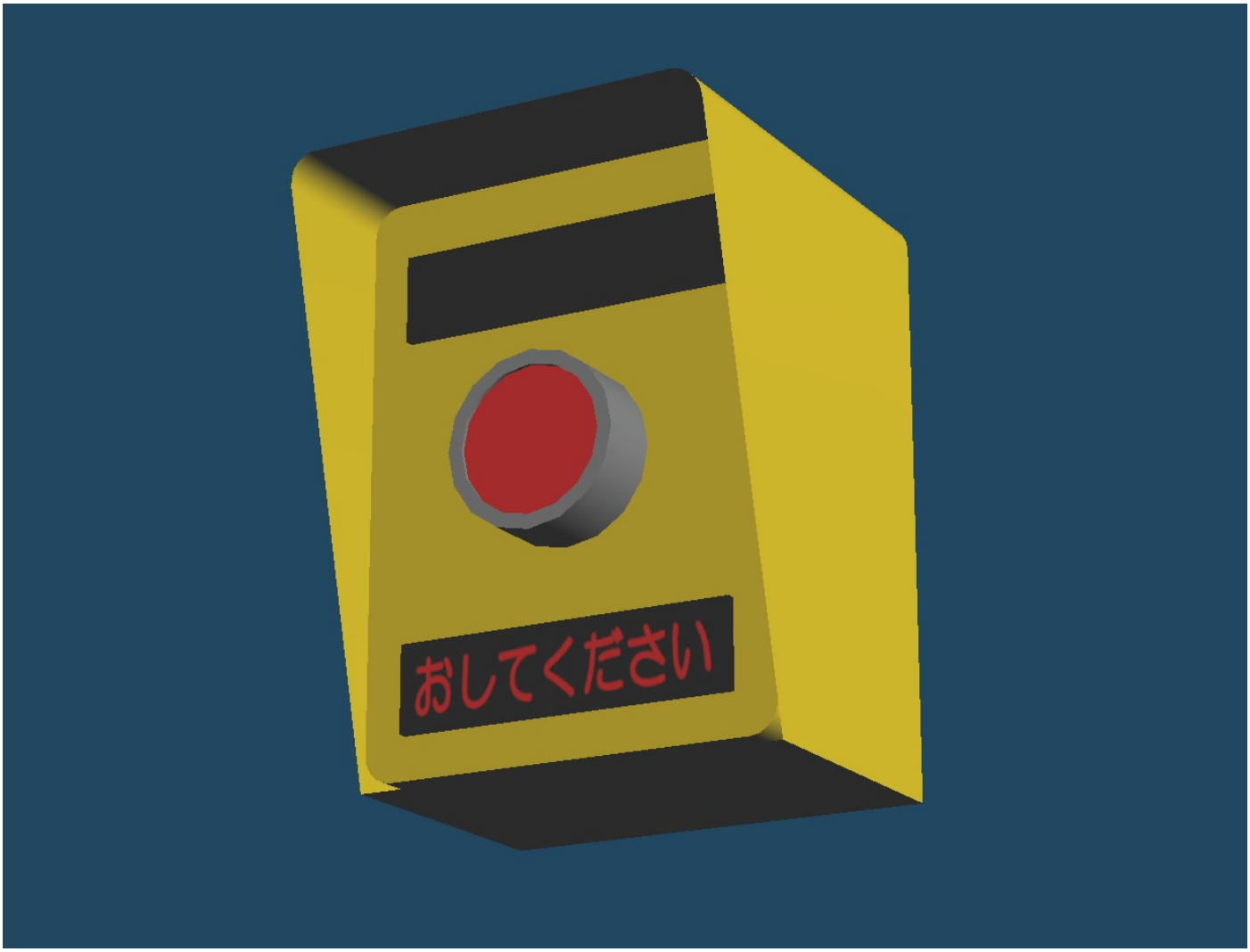
3-5: 設置終了

あとはつないだ瞬間勝手に動くはずです。動かない場合は後述するトラブルシューティングを参照してください。なお制御機を後から移動することはできない（一度破壊して設置しなおし・設定しなおしになる）ので、場所は慎重に設定してください。また、Mod の仕様上何度も何度も同じ場所に制御機を設置しては破壊して…を繰り返すと、過去の設定が怨霊のごとくよみがえってくることがあります。

↑底面は手抜きしているのでできるだけ地面に設置してご利用ください。

4: 押ボタン箱の使い方

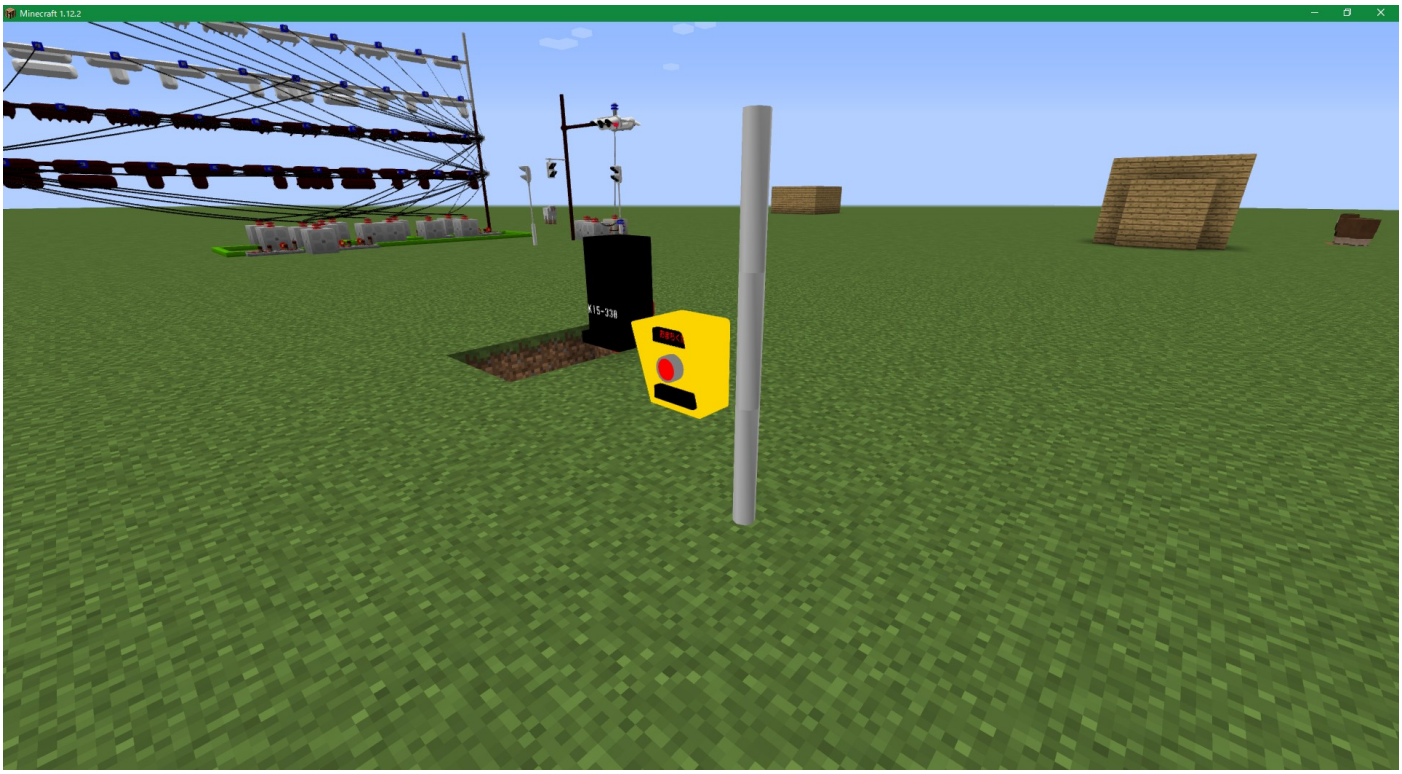
Ver0.2β より追加された**押ボタン箱**の使い方について説明します。押ボタン箱は、押すことによって押ボタン箱を中心とした球体半径 32 マス以内の制御機全てに検知信号を送信します。検知範囲に複数の制御機がある場合は、最も近い制御機にアタッチします。なお、**制御機を破壊する際は必ず押ボタン箱を破壊してから制御機を破壊するようにしてください**。さもなくば、押ボタン箱のボタンを押した際に高確率でぬるぽクラッシュします（この判定は Mod 内ではしづらいのでなるべく守ってください）。



押ボタン箱は MQO ファイルを使用して作成しています。押ボタン箱を自作して中身をすり替えることによって、自作した押ボタン箱を使用することができます。Mod 内での変更（RTM のようにモデルを選択できる機能）はサポートしていません。また押ボタン箱をすり替える場合は、座標の正規化をしないといけません（0-1 の間で座標をそろえてください。1 ブロックの座標が 1 となります）。詳しくはドキュメントの範疇を超えるのでわかんなかったら GitHub に公開している「GingaCore」の MQO クラスをご覧ください。意味が分かんない場合はもうすり替えるのをあきらめてください。そういうつもりで作ったわけではないので…。

4-1: 設置する場所

設置はどこにでも行うことができますが、フェンスや電柱に設置すると残念なことに押ボタン箱が浮いてしまいます。



また、制御機から直線距離で 32 ブロック以内に設置しないと、迷子の押ボタン箱になります。迷子になった押ボタン箱は、一度押すと永遠に押せなくなるので撤去必須です。また、アタッチした制御機が「押ボタン式信号機」やカスタムスクリプトで「detected」を使うものだった場合以外ではやはり永遠に押せなくなります。撤去して設置しなおしましょう。

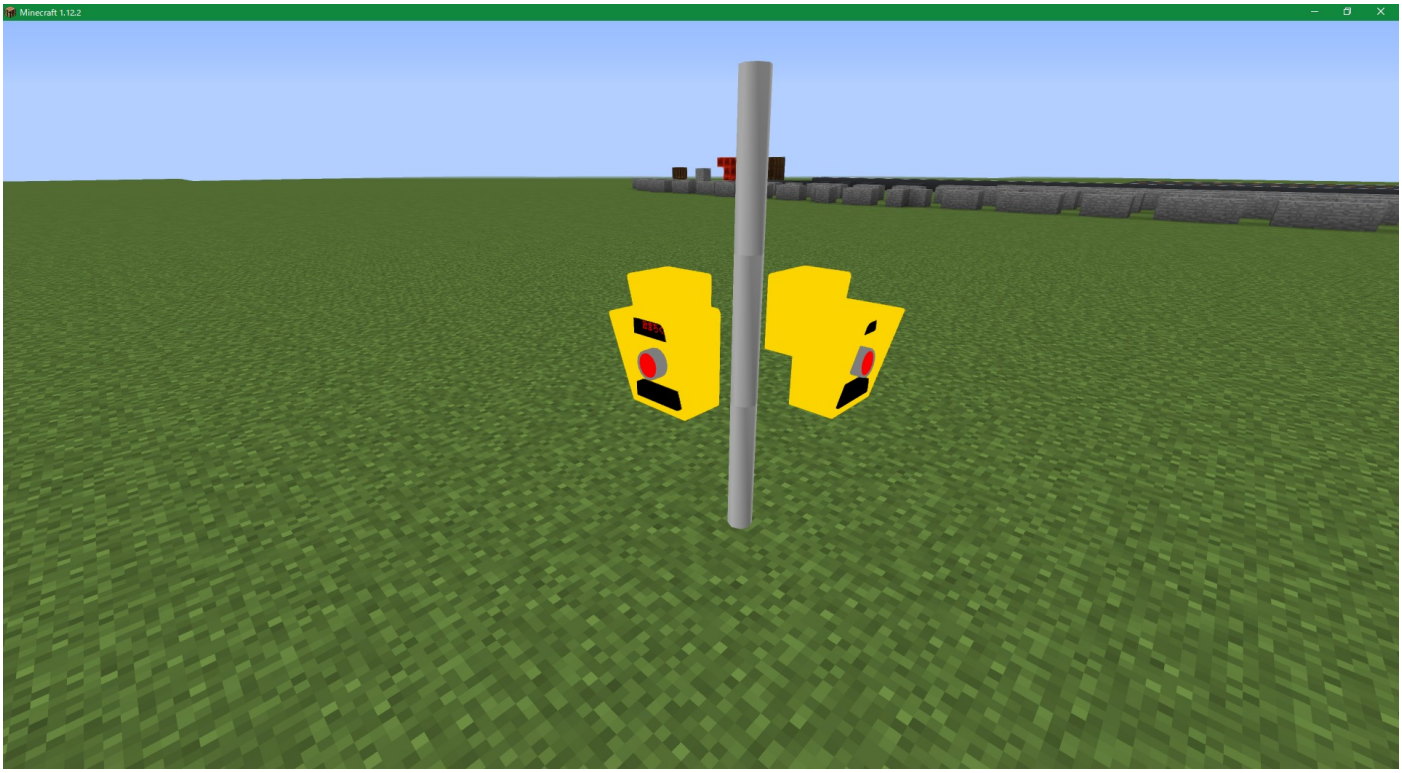
迷子になっているか確認する方法が現時点では実質ありません。32 ブロック以内かどうか怪しい場合は近くに設置してみましょう。次期バージョンではこの値もコンフィグで変えることができるようにするつもりですが、あまり増やしすぎると関係ない制御機にまでアタッチしてしまいます。

4-2: 押した後の挙動

1 回押すと、制御機をアタッチしている場合は「おまちください」となり、その後歩灯が青になったら全 LED が消灯します。その後、点滅すると LED が再点灯します。ここまでは押すことができません。「おしてください」の表記になった後に押すことができます。カスタムスクリプトの場合は適切に終了処置を行わないと永遠に押せなくなります。

また、同じ制御機にアタッチしている複数のボタンがある場合、半分連動します。半分、というのは、押された瞬間だけは反応しないからです。つまり、ボタン A とボタン B がどちらも制御機 C を参照していたとして、ボタン A を押したらボタン A だけが「おまちください」となります（現実ではボタン B も切り替わりますがそこまで対応できません）。その後、制御機 C が切り替わるとどちらのボタンも消灯します（ここからは連動）。なので半分ということになります。

向きは 4 方向のみです。斜めを向かせることはできません（やろうとすればできるんでしょうが、OpenGL の知識がないため開発者は匙を投げました）。



このように 4 方向の向きに変えることができます。

5: トラブルシューティング

Mod を導入した後に Minecraft がクラッシュする場合、起動後にワールドを開くと落ちる不具合など、当 Mod によるクラッシュに対する症状と対処方法を載せておきます。利用者から情報提供をいただいたもの、開発環境で再現したもの、予知できるものを載せていますが、必ずしもすべての現象を載せられるわけではありません。また、（ないとは思いますが）他の Mod との競合によってエラーが起きている場合は大体こっちが悪いので、GTC を無効化すれば起動できます。

なお後ほど利用規約にも記しますが、当 Mod を導入したことでワールド消えたりパソコンが壊れたり人生が詰んだりしても、開発者は一切の責任を負いませんのでご了承ください。

5-1: 起動すらしてくれない

Minecraft のタイトル画面に出る前に落ちる場合は、まず GTC を抜いて起動してみてください。それで解消する場合、落ちた際のクラッシュレポートを用意してください。それをもとに、後述するクラッシュレポートの例外をお読みください。

5-2: 起動するけど読み込まれていない

Minecraft は起動したが、Mod リストを見ても「GingaCore」「GTC」がない場合です。この場合は、Mods フォルダに入っていません。入れる場所を間違えているため、正しい場所に入れてください。RTM と同じ場所です。1 階層でも違うと読み込んでくれません。

5-3: 制御機を設置すると落ちる

ワールドに入って制御機を設置するとクラッシュする場合があります。2 パターンありまして、1 つは「Can't connect Server（接続を維持できません）」と出て強制的にサーバーダウンしてタイトルに戻ります。この場合はパケット通信関連のエラーが起きているはずです。出力ログの「netty」関連でエラーが量産されていると思います。これは大体こちら側のミスなので、お手数ですがスクショなりで出力ログの該当する部分（なんたら Exception から at ～ and 444 more...みたいなところまで）をお問い合わせください。

もう 1 つは、クラッシュレポートが生成されてダウンする場合です。この場合は、RuntimeException 関連のことが多いため（ランチャーも落ちることがほとんど）、後述するクラッシュレポートの例外をお読みください。

5-4: 制御機を更新すると落ちる

だいたい 5-3 と同じです。5-3 の対処をしてください。

5-5: 押ボタンを押すと落ちる

押ボタン箱のボタンを押した瞬間落ちる場合、その押しボタンがアタッチしている制御機が存在しなくなったか、データが壊れた可能性があります。逆を言えば押すまでは落ちないので、そのようになった場合は元あった場所に制御機を置くか、押ボタン箱を撤去して再度設置しなおす必要があります。

5-6: ワールドを開くと落ちる

新規ワールドを作った際や、既存のワールドで GTC を入れて開いたことがない場合、新規オープン時にまれに落ちます。最近のバージョンではこのようなことはめったにありませんが、原因は GTC が利用するワールドデータが正常に生成されなかったことがほとんどです（その証拠にクラッシュレポートもだいたい NullPointerException）。この場合は、一度ワールドを作り直すと解決します。ちなみに Ver0.2β では何回やっても落ちるのでこのバージョンの使用は避けてください。

5-7: OpenGL エラーが出る

まずないですが、ごくまれにログに OpenGL 関連のエラーが出たり、押ボタンや制御機に支店を合わせたときに空が一面に広がることがあります。Minecraft を再起動すると直ることがあります。もしくはワールドバックアップ取って、視点が消える瞬間のところで左クリックして制御機なり押ボタン箱を壊してみてください。

5-8: 信号が固まっている

Optifine を導入しているとたまに固まることがありますが、これは仕様です。チャンクローダーとか使ってどうにかすれば多分解決します。

また、いつまでたってもサイクルが切り替わらない場合は、何らかの理由で tick が壊れ、サイクルの終了ができずに無限ループに陥っている可能性があります。この場合、int 値上限値の 2147483647tick（約 1 億秒）待つとクラッシュしますが、20 年単位で待つ人なんていないと思うので、無限ループに陥ってしまった場合は、以下の順番で実行してみてください。

1. 制御機を夜間点滅にして、強制的に夜にしたあと、もう一度朝にする。
2. これで直らない場合は、制御機のタイプを押ボタンに変えてしばらくしたら戻す。
3. これでもダメな場合は、一度制御機を壊して設置しなおす（コネクタは消えない）。

これも再現不可能（突如起こる）バグなのでどうしようもありません。この対処で一時的にしのいでください。

6: クラッシュレポート対応表

クラッシュレポートが出て落ちてしまった場合、そのクラッシュレポート各々に対する「原因」と「対処」を記しました。

6-1:「例外」について

Java の話になるのでわかんない方は軽く読んでください。例外とは、その名の通り、「本来想定していない何かが起こった」ことを指します。例外は英語で「Exception」と記します。Java の例外も「～Exception」であることが多いです（まれに「～Error」がありますがあれは例外ではなくエラーです、ただしここでは同一視します）。これから頻繁に「例外」が出てきますが、例外と言ったら「～Exception」だと思ってください。例えば「ぬるぽ」は「NullPointerException」です。

6-2: 例外の探し方

例外は、クラッシュレポートの上の方に書いてあります。最初の方のヘッダー文はどうでもいいところなので関係ありません。最後のほうにある「System Details」も動かしているシステム（パソコン）の情報なので、トラブルシューティング自体には必要ありません。ただし開発者に問い合わせする場合は全て送っていただく必要があります。

以下に、クラッシュレポートの一部を掲載します（ドキュメントが埋まるのでスクショです）

多分ほとんどの方はこれを見ただけで吐き気を催すのではないでしょうか。まあ下手な論文より読むのが厄介ですし、わかります。でもここで見たいのはほんの一部なんです。どこを読むかというと、このクラッシュレポートで言えば次の部分です。

```
   Description: Updating screen events

java.lang.NullPointerException: Updating screen events
    at jp.gingarenpo.gtc.gui.GUIControl.func_146284_a(GUIControl.java:297)
    at net.minecraft.client.gui.GuiScreen.func_73864_a(GuiScreen.java:443)
    at net.minecraft.client.gui.inventory.GuiContainer.func_73864_a(GuiContainer.java:323)
    at jp.gingarenpo.gtc.gui.GUIControl.func_73864_a(GUIControl.java:179)
    at net.minecraft.client.gui.GuiScreen.func_146274_d(GuiScreen.java:533)
    at net.minecraft.client.gui.GuiScreen.func_146269_k(GuiScreen.java:501)
    at net.minecraft.client.Minecraft.func_71407_l(Minecraft.java:1759)
    at net.minecraft.client.Minecraft.func_71411_J(Minecraft.java:1098)
    at net.minecraft.client.Minecraft.func_99999_d(Minecraft.java:398)
    at net.minecraft.client.main.Main.main(SourceFile:123)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:497)
    at net.minecraft.launchwrapper.Launch.launch(Launch.java:135)
    at net.minecraft.launchwrapper.Launch.main(Launch.java:28)
```

だいぶシンプルになりました。これが例外と呼ばれるものです。まず、1 行目を見て下さい。1 行目にはどんな感じのエラーか簡易的な説明があります（実質 2 行目の例外内容です）。この場合は「Updating screen events」、つまり GUI 関連のエラーだとわかります。

次に、2 行目を見て下さい。「java.lang.**NullPointerException**」がありますね。つまりぬるぽが発生していることになります。

3 行目以降は、その例外がソースコードのどこで発生したかを記しています。「at」となっているのは「ここで発生したよ」ってことです。いっぱいあるのは、「ぬるぽを呼び出した 3 行目を呼び出したのは 4 行目で、4 行目を呼び出したのは 5 行目で…」といったように呼び出し元をとにかく列挙しているからです。後ろの方は正直関係ありません。重要なのは大体上から 3~5 つくらいです。

この行で、2 行目を「例外」、3 行目以降を「発生個所」としています。発生個所の at の後にはパッケージ名が書かれています。だいたいここに Mod 名の一部が隠れているので（詳細は割愛）、ここを読んでみると 3 行目が「jp.Gingarenpo.**gtc**.…」となっています。ここからもわかるように、このエラーは GTC が引き起こしていることになります。

ドットの最後にある括弧の手前は関数（メソッド）名ですが、「func_99999_d」とかなっているのは、難読化されていて読み解くのが困難とされています。Mod 開発経験者以外は深入りすると沼にハマるので無理せず開発者にお問い合わせください。

それでは、自身のクラッシュレポートは読み解けたでしょうか？　読み解いた例外と発生個所をメモするなり覚えたら、以下のクラッシュレポート対応表をご覧ください。ただし、ここに載っていない例外も発生する可能性があります。その際はお問い合わせください。自力で解決できるようなら（そして解決したら）ご報告いただけると助かります。

6-3: 対応表

再現しているクラッシュレポートが非常に少ないため、利用者の皆様からのクラッシュレポートも参考にしたいと思っています。わからないことがあれば一緒に解決しますのでお気軽にお問い合わせください。お問い合わせ方法は末尾に記します。

例外	発生個所	原因	対処
`NoSuchMethodError`	どこでも	クライアント（お使いのパソコン）の難読化に当 Mod が対応していない、GingaCore が最新版じゃない	GingaCore を最新版にアップデートして、再度実行してください。それでも同じエラーが出る場合は、当 Mod はおそらくご利用いただけません。アップデートにより解消する場合もあるのでお問い合わせいただくのもありです。
`ClassNotFoundException`	どこでも	GingaCore が最新版じゃない	GingaCore を最新版にアップデートしてください。
`NullPointerException`	jp.gingarenpo.gtc.gui～	想定外の入力をされた可能性が高い	数値を入力したかどうか確認してください。それでもうまくいかない場合はお問い合わせください。大体こちらのチェックミスです。
	jp.gingarenpo.gtc.data～	ワールドの生成に失敗している	もう一度ワールドを作成してみてください。それでもうまくいかない場合はお問い合わせください。
`IndexOutofBoundException`	だいたいが jp.gingarenpo.gtc.network～	パケットの送受信に失敗した	何度かトライしてもダメな場合はお問い合わせください。

7: カスタムスクリプトについて

それでは、最後にカスタムスクリプトの作り方に関して解説します。現在この内容はβ段階であり、今後頻繁に仕様変更がされる可能性があります。ご了承ください。

カスタムスクリプトを作成する場合は、現在 JavaScript の知識が必要となります。…と言っても、あまり特有のメソッドは使用しないことがほとんどだと思います（凝ったことをしたい場合は必要です）。RTM のモデルパックで JavaScript をいじったことがある方はほとんど同じなので、着手しやすいと思われます。

カスタムスクリプトを作成すると、組み込みの制御以外にも凝ったことができるようになります。矢印付き制御や、歩車分離式の制御なら簡単に作れます。

それでは、カスタムスクリプトの仕様についてまずは記します。

7-1: カスタムスクリプトの仕様

まず、JavaScript であることが条件です。Ver0.4β 現在、独自書式には対応していません。対応次第追記します。拡張子は何でもいいですが、「.js」を使うことをお勧めします。また、スクリプトに日本語を含んでも構いません。ただし、文字コードはどんな状態でも「UTF-8」をご利用ください。UTF-8 以外では文字化けする可能性が高く、コンパイルに失敗します。従って、メモ帳などで作成するのはお勧めしません。Notepad++とかその辺をお使いください。Android だったら Jota+とかお勧めですがまあスマホでは導入できないのであまり意味ないというか。

JavaScript として有効なあらゆるメソッド・関数を使用することができます。独自で定義することもできます。ただし定義しただけでは使用されません！

また、1 つだけ必ず実装してほしいメソッドがあります。「onStart」というメソッドです。ここには、初期状態（サイクルカウント 0）の信号を初期化する際に使用しますので、そのコードを書いておいてください。

サイクルの内容は、トップレベルメソッドで OK です。

7-2: 使用可能な組み込み変数・関数一覧

JavaScript では、それ標準で使用可能な変数や関数以外に、GTC が提供するいくつかの変数と関数を利用することができます。使用可能なものをいかに列挙します。

7-2-1: 変数

以下の変数は、グローバル変数として使用可能です。ただし、値を変更しても反映されません。

変数名	内容	変数タイプ・備考
greenTime	制御機の青時間	整数
yellowTime	制御機の黄色時間	整数
redTime	制御機の赤時間	整数
allRedTime	制御機の全赤時間	整数
optionValues	制御機のカスタムデータ	整数の配列
detected	検知中かどうか	論理型（検知していれば true）
tick	サイクル開始後の経過 Tick	整数
GREEN	青信号レベル（Config）	整数、Ver0.4β 現在未実装
YELLOW	黄色信号レベル（Config）	整数、Ver0.4β 現在未実装
RED	赤信号レベル（Config）	整数、Ver0.4β 現在未実装
NONE	無信号レベル（Config）	整数、Ver0.4β 現在未実装

7-2-2: オブジェクトとメソッド

操作可能なオブジェクトとして、制御機の一部の値の書き換えを可能とする「control」というオブジェクトが提供されます。このオブジェクトに set～メソッドを使用することにより、数値を書き換えることができます。ほとんどは読み込みもできます。

メソッド名	内容	戻り値・備考
setSignal(int)	主道車灯の信号レベルを指定します。	ハードコーディングは避けて、できるだけ変数をご利用ください（Config で弄れる内容を定数のようにして代入しています）。
setSubSignal(int)	側道車灯の信号レベルを指定します。	
setPVSignal(int)	主道歩灯の信号レベルを指定します。	
setSubPVSignal(int)	側道歩灯の信号レベルを指定します。	
setDetected(boolean)	検知信号を受信しているかを指定します。押ボタンなどの検知終了時に使用します。	通常は false にする用途で使用します。
setTick(0)	Tick を変更します。サイクル終了時に使用します。	0 以外指定しないでください。

7-2-3: importPackage()について

この魔法を唱えると、GingaCore のメソッドとかにアクセスすることができます…が、まず使用することがないのでこの欄は無視して結構です。

一応、GingaCore のパッケージは「jp.gingarenpo.api」となっています。

7-2-4: Java に値を渡すことは可能か？

できません。正確には、与えられた「control」で設定できる内容のみが渡せます。独自の値を Java に渡すことはできません。そんなことをしても Java 側で取得するコードを書いていません。

7-3: 簡単なサンプルと解説

基本的に、JavaScript の内容は 1Tick 毎に呼び出されます。その際に、tick 変数にサイクル開始からの経過 Tick が渡されるため、これをもとに if 分岐するだけでカスタムスクリプトは完成してしまいます。高度なことをやりたい場合は頑張ってください。

＜サンプルとして、ただ交互に点滅するだけのスクリプト＞

```
if (tick <= 10) {
 // 1秒 = 20Tickなので0.5秒間
 control.setSignal(YELLOW); // 黄点滅
 control.setSubSignal(NONE); // 主道が点灯している間は側道は消灯
} else if (tick <= 20) {
 // さらに0.5秒間
 control.setSignal(NONE); // 側道が点灯している間は主道は消灯
 control.setSubSignal(RED); // 赤点滅
} else {
 // 1サイクルを終えたのでループ
 control.setTick(0); // サイクル終了を判断するメソッド
}

// ↓必ず記す必要がある
function onStart() {
 // 初期状態の信号レベルを指定する
 control.setSignal(YELLOW);
 control.setSubSignal(NONE);
 control.setPVSignal(NONE); // 歩灯は常時消灯
 control.setSubPVSignal(NONE); // 上に同じく
}
```

こんな感じでスクリプトを書けば動きます。よくわかんない場合はいろいろ弄ってみるといいかもしれません。

なお、文法的におかしい場所があると、事前コンパイルに失敗して Minecraft がクラッシュします（ScriptException が発生します）。配布などをされる際は事前にコンパイルエラーが起きないか確認してください。~~開発者は確認放棄しています~~。

あとがき

利用規約とコンフィグに関してはまだ記していませんが、おそらく誰もこのドキュメントを書いている時点でダウンロードしていないと思うので、おいおい書いていくことにします。

当 Mod を利用して少しでも誰かの役に立つのであれば、幸いです。

銀河連邦

お問い合わせについて

お問い合わせの場合は、以下の Twitter アカウントか GMail までお送りください。

Twitter: https://twitter.com/Ginren_Factory (@Ginren_Factory、銀連製作所公式アカウント）

Twitter: https://twitter.com/Gingarenpo (@Gingarenpo、銀連製作所代表兼 Mod 開発者）

Gmail: gingarenpo.9844@gmail.com (反応は遅れます）

些細なことでもお問い合わせいただければ対応いたしますので、迷ったらお気軽にどうぞ♪

※明らかに意図の伝わらないものは無視することがあります

※Twitter はなるべく DM 機能をお使いください。リプは無視することがあります。