シンセサイザー入門

Naomasa Matsubayashi

自己紹介

はじめまして、おひさしぶりです。松林です。★Twitterでは@fadis_とかいう名前でやっています。

松林 尚理

自己紹介

はじめまして、おひさしぶりです。松林です。★Twitterでは@fadis_とかいう名前でやっています。

松林 尚理

Twitter

@fadis_

自己紹介

はじめまして、おひさしぶりです。松林です。★Twitterでは@fadis_とかいう名前でやっています。

松林 尚理
githubはじめました

https://github.com/Fadis/

自己紹介

最近githubも始めて、★今回紹介するコードもこちらで公開するのです。

松林 尚理
githubはじめました

https://github.com/Fadis/

今回紹介するコードも
ここで公開中

自己紹介

最近githubも始めて、★今回紹介するコードもこちらで公開するのです。

古の昔

今を去る事7ヶ月前、第一回カーネル/VM探検隊 関西がここ京都で行われました。★その時、1bit Musicというタイトルで発表をさせていただきました。★この発表では、beep1音だけでどんな音色を奏でる事ができるかについてお話ししましたが、

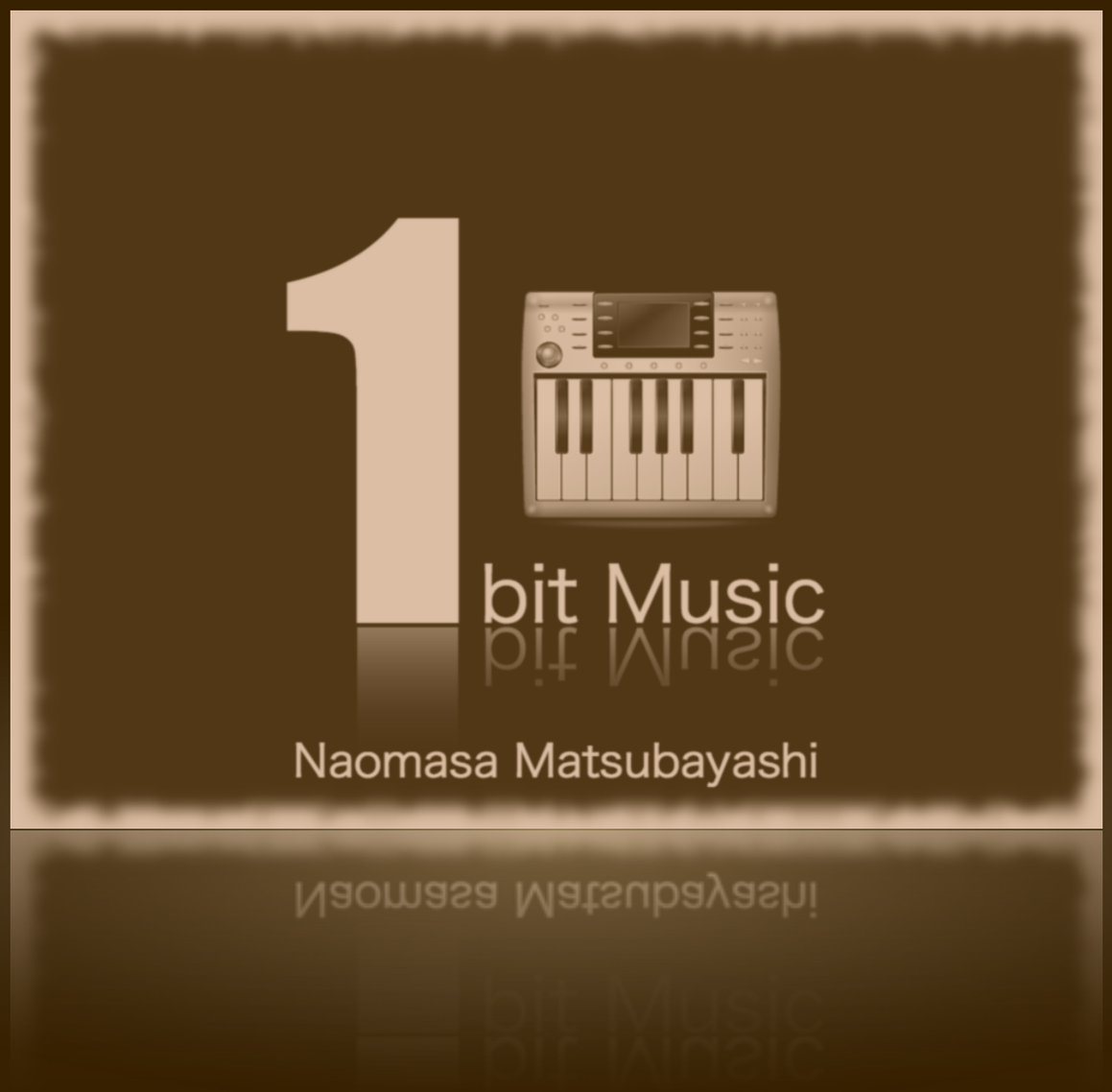
古の昔

第一回カーネル/VM探検隊@関西

今を去る事7ヶ月前、第一回カーネル/VM探検隊 関西がここ京都で行われました。★その時、1bit Musicというタイトルで発表をさせていただきました。★この発表では、beep1音だけでどんな音色を奏でる事ができるかについてお話ししましたが、

古の昔

第一回カーネル/VM探検隊@関西



今を去る事7ヶ月前、第一回カーネル/VM探検隊 関西がここ京都で行われました。★その時、1bit Musicというタイトルで発表をさせていただきました。★この発表では、beep1音だけでどんな音色を奏でる事ができるかについてお話ししましたが、

古の昔

第一回カーネル/VM探検隊@関西

beep1音で何が出来るか

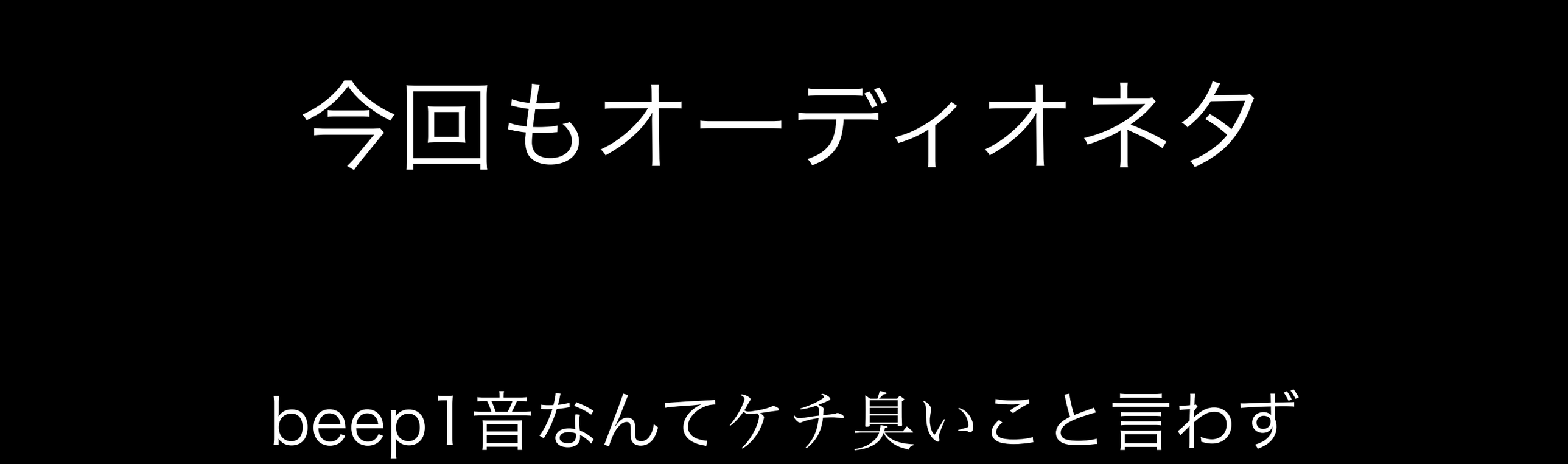
今を去る事7ヶ月前、第一回カーネル/VM探検隊 関西がここ京都で行われました。★その時、1bit Musicというタイトルで発表をさせていただきました。★この発表では、beep1音だけでどんな音色を奏でる事ができるかについてお話ししましたが、

今回もオーディオネタ

今回も引き続きオーディオネタです。★しかし今回は、beep1音などというケチ臭いことは言わず、★ガチでシンセサイザーを作ってみようと思います。

今回もオーディオネタ

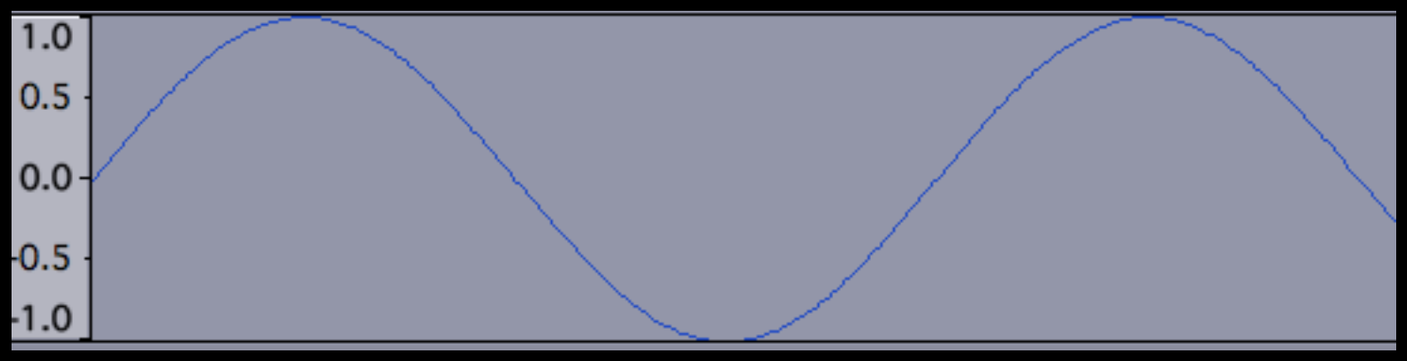
beep1音なんてケチ臭いこと言わず

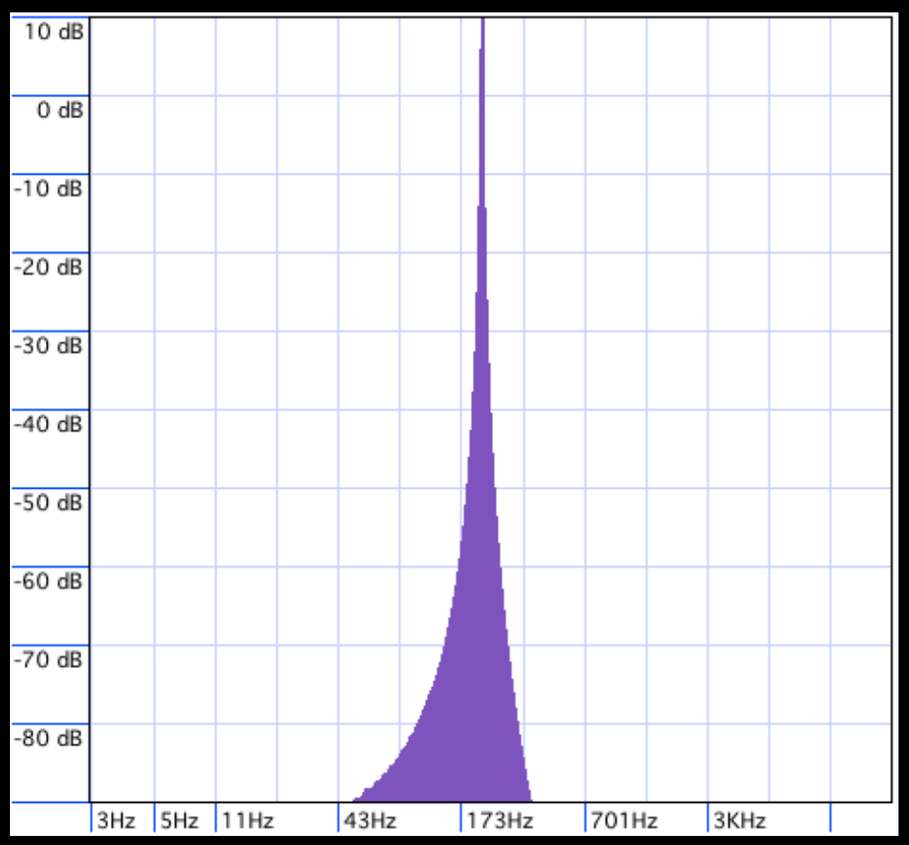


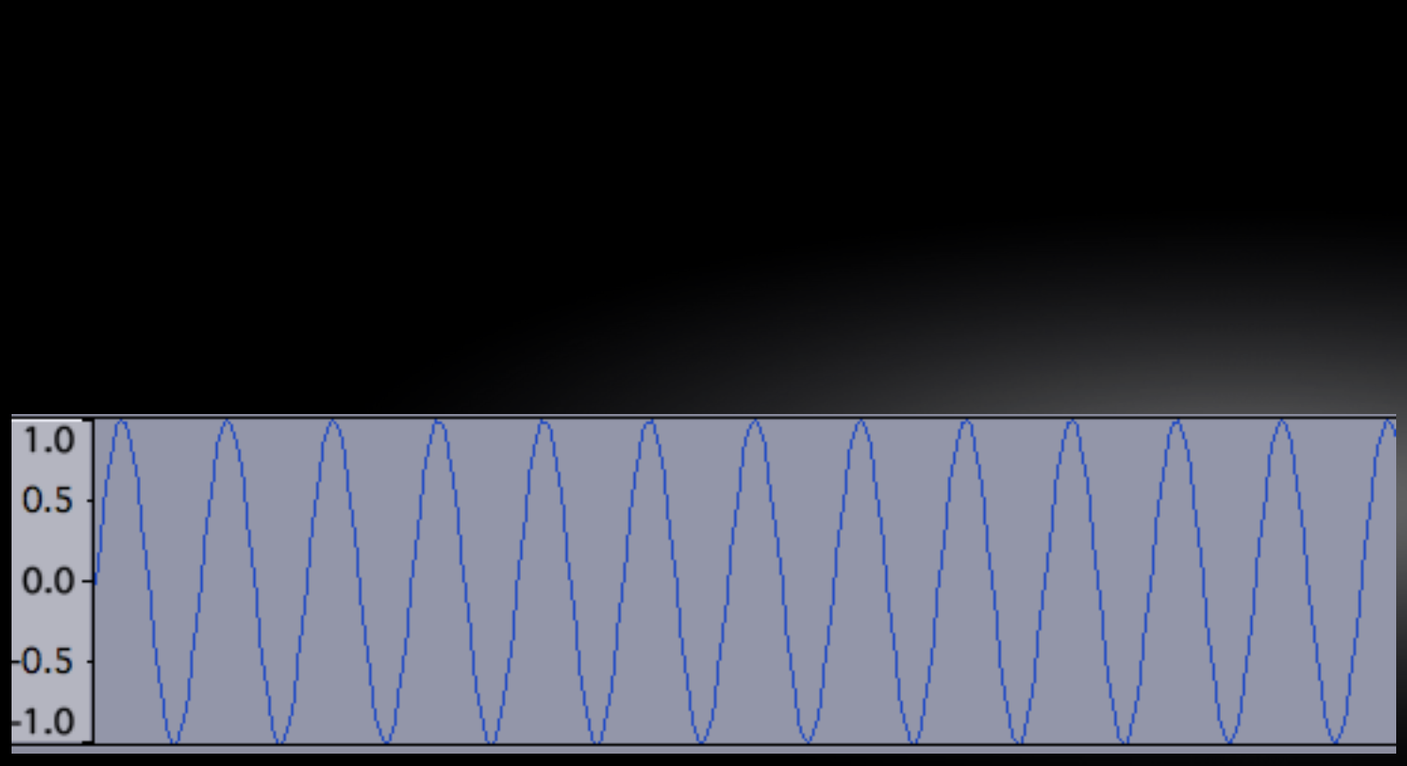
今回も引き続きオーディオネタです。★しかし今回は、beep1音などというケチ臭いことは言わず、★ガチでシンセサイザーを作ってみようと思います。

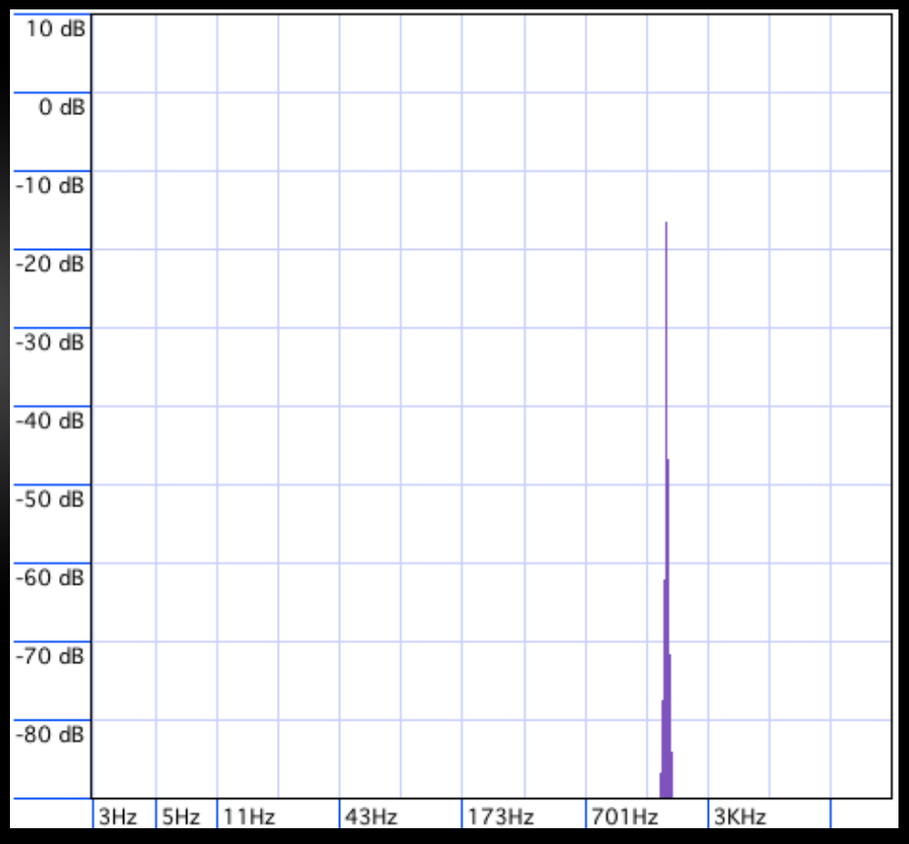
今回も引き続きオーディオネタです。★しかし今回は、beep1音などというケチ臭いことは言わず、★ガチでシンセサイザーを作ってみようと思います。

音の周波数によって音階が変わる



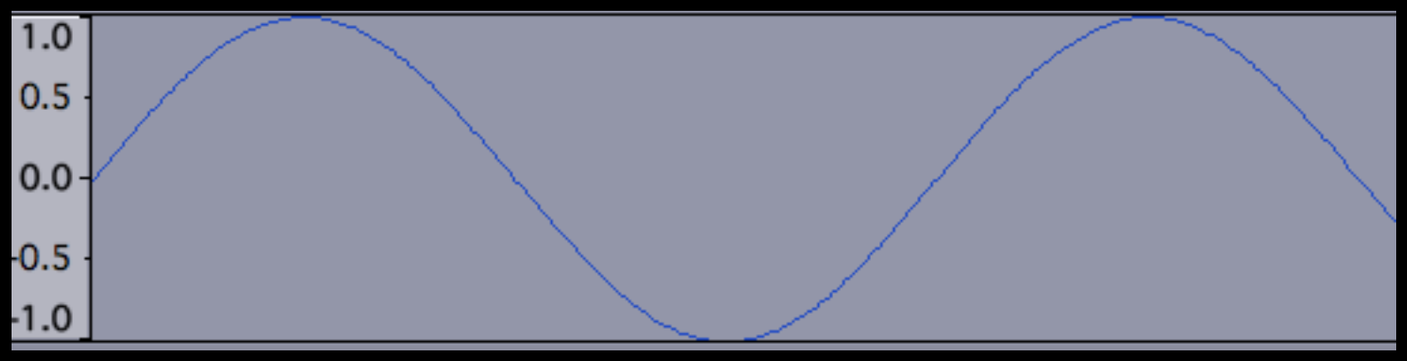


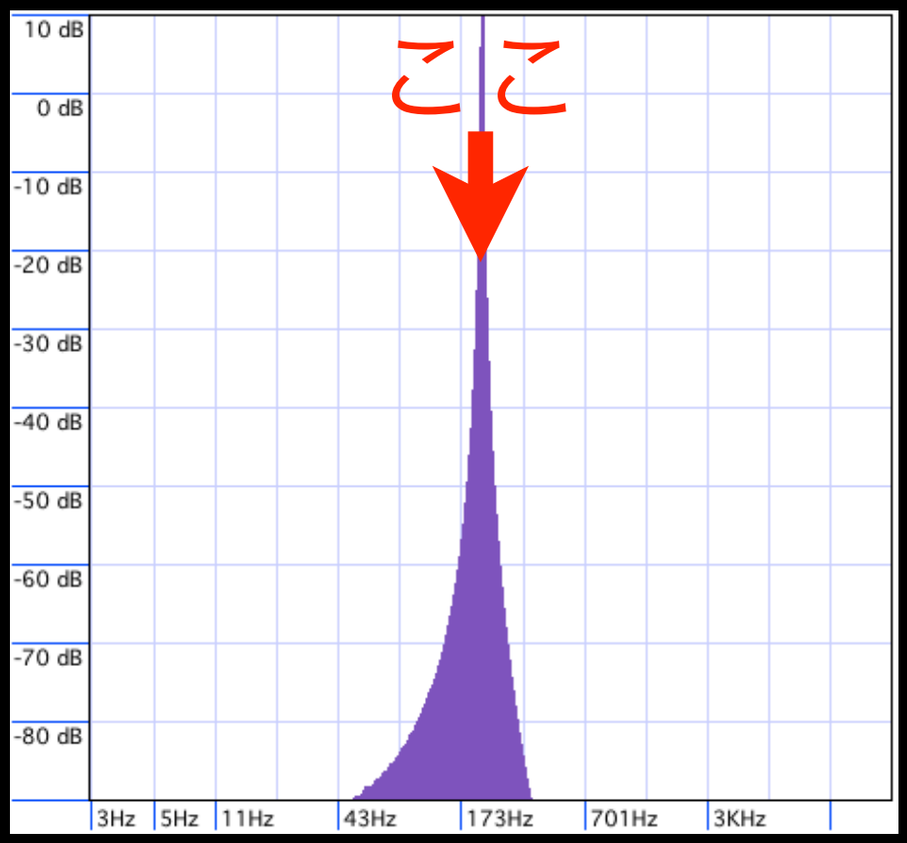


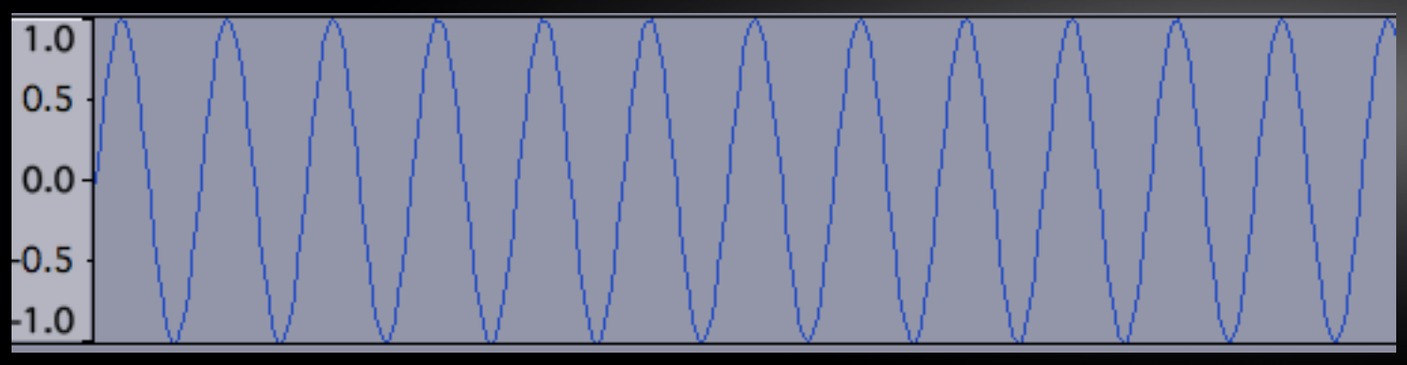


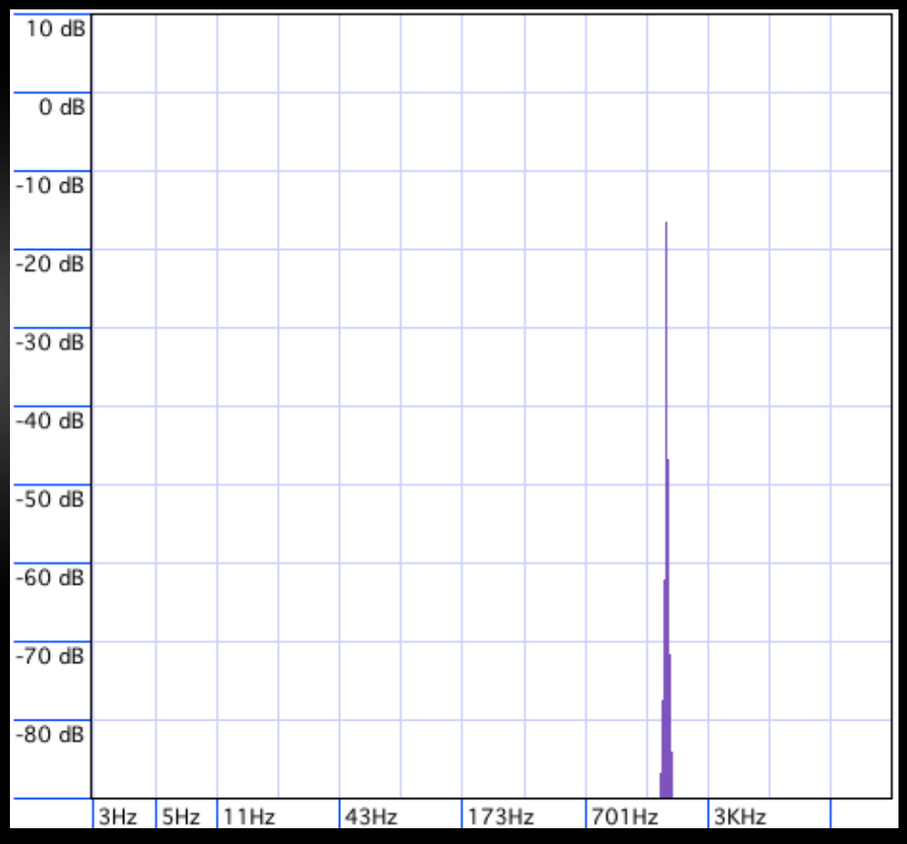
まず皆さんご存知の事かと思いますが基本的な事柄について確認しておきましょう。音にはドレミファソラシドといった音階があり、音楽を演奏する為にはこれらを出し分ける必要があります。音階の違いとは音の周波数の違いです。例えば、サイン波でラの音を出すと、その周波数のピークは★このあたりになるのですが、1オクターブ高いラの音を出すと、その周波数のピークは★このあたりに移動します。

音の周波数によって音階が変わる



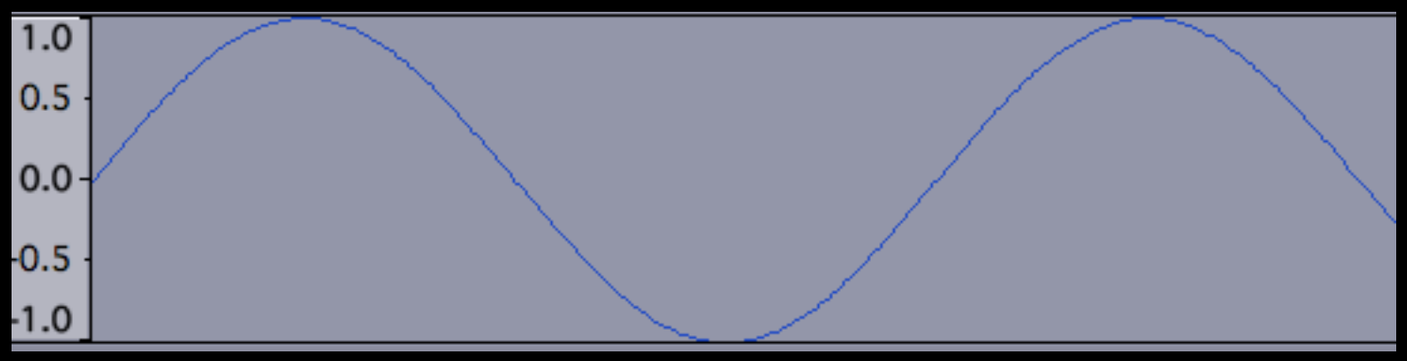


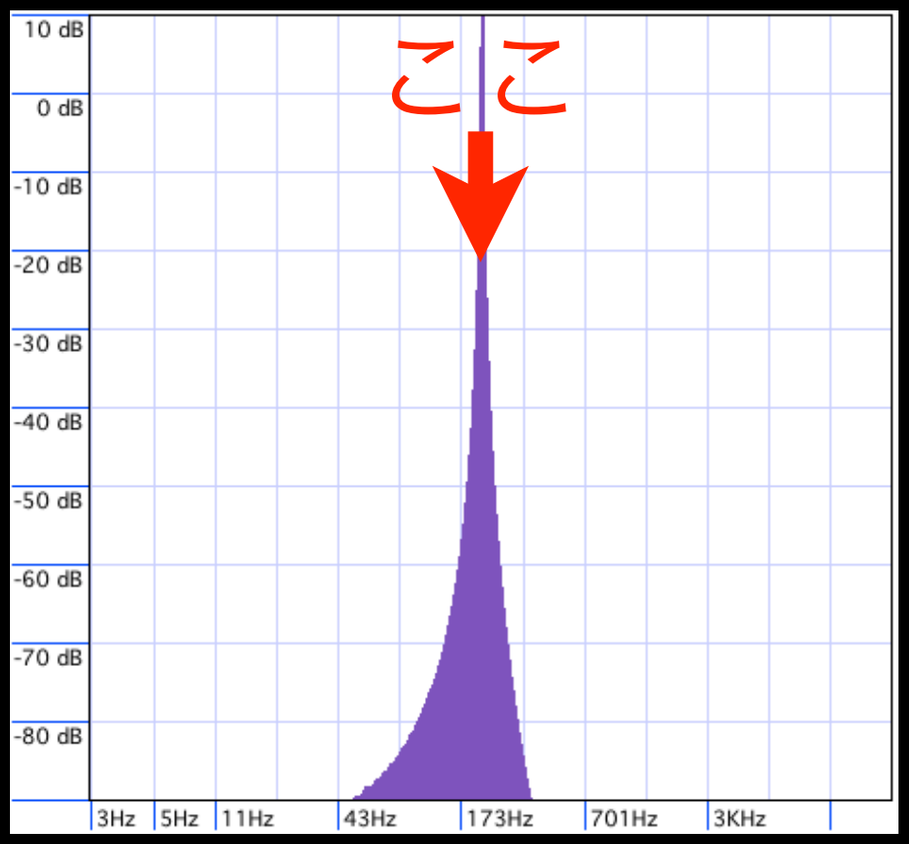


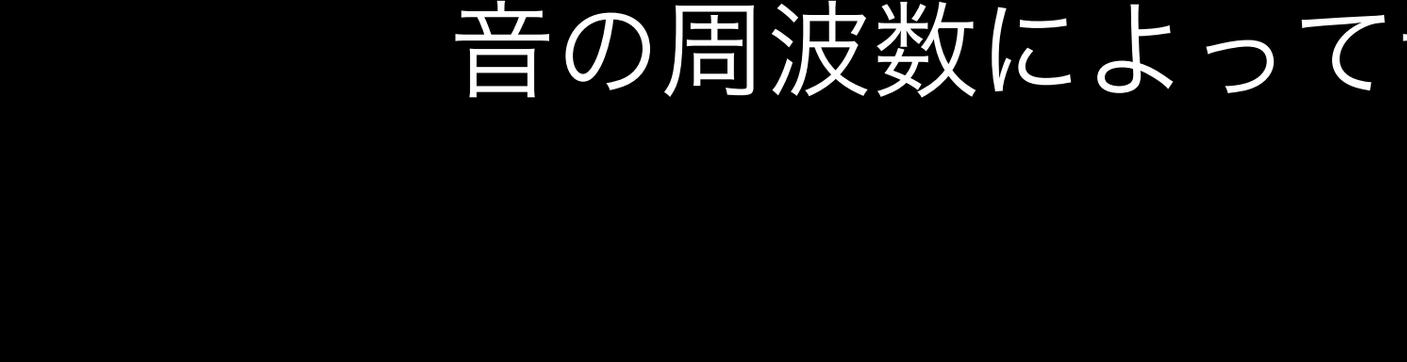


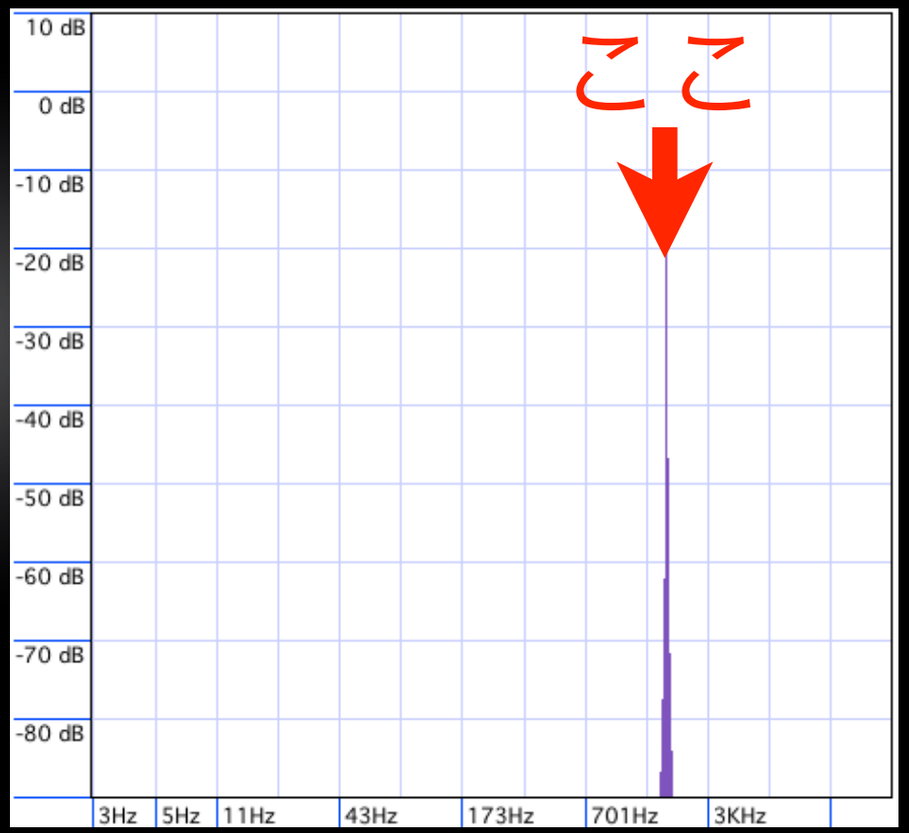
まず皆さんご存知の事かと思いますが基本的な事柄について確認しておきましょう。音にはドレミファソラシドといった音階があり、音楽を演奏する為にはこれらを出し分ける必要があります。音階の違いとは音の周波数の違いです。例えば、サイン波でラの音を出すと、その周波数のピークは★このあたりになるのですが、1オクターブ高いラの音を出すと、その周波数のピークは★このあたりに移動します。

音の周波数によって音階が変わる



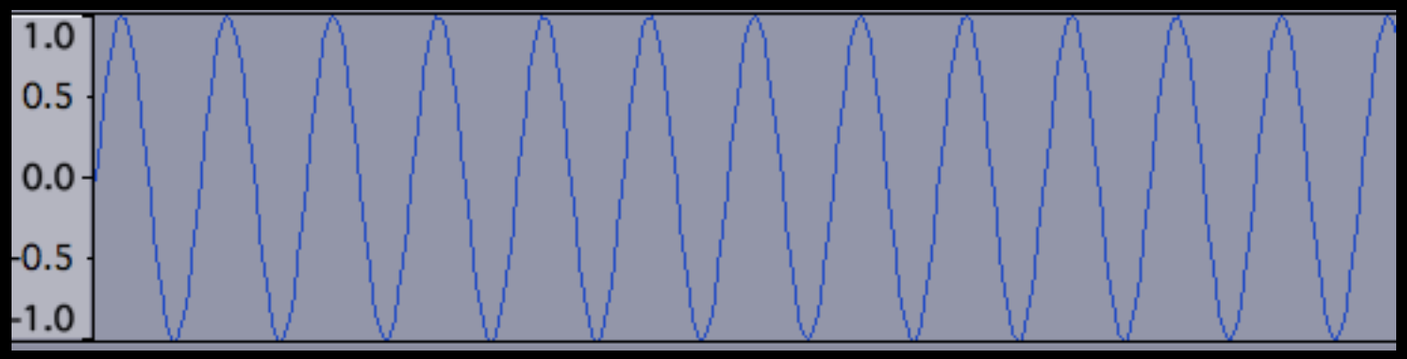


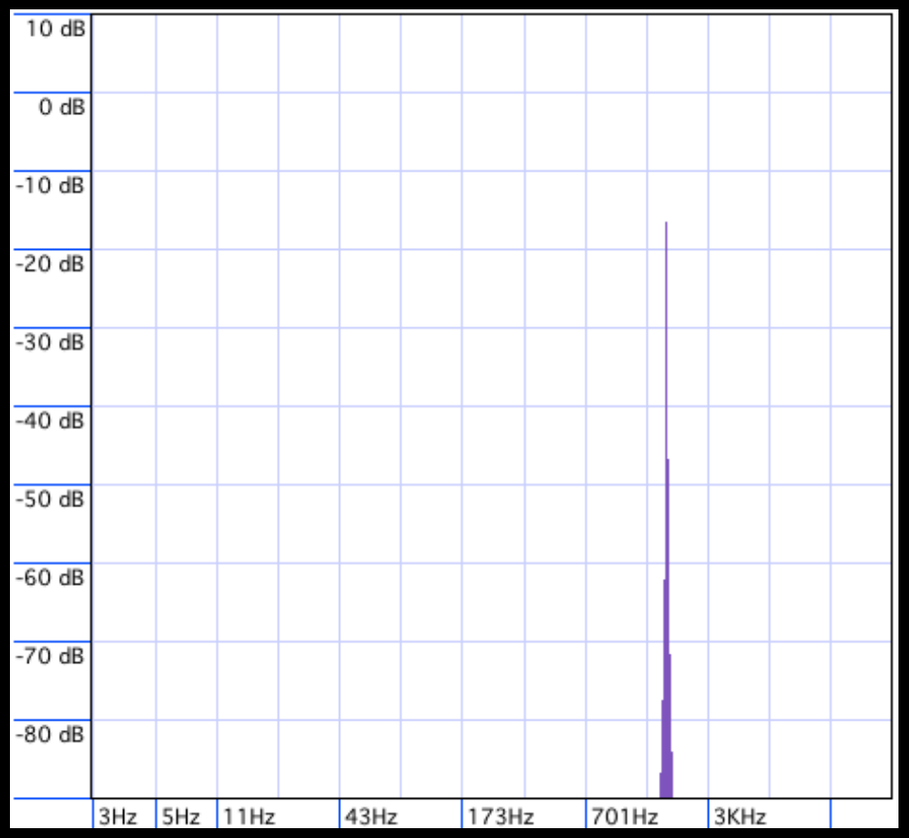


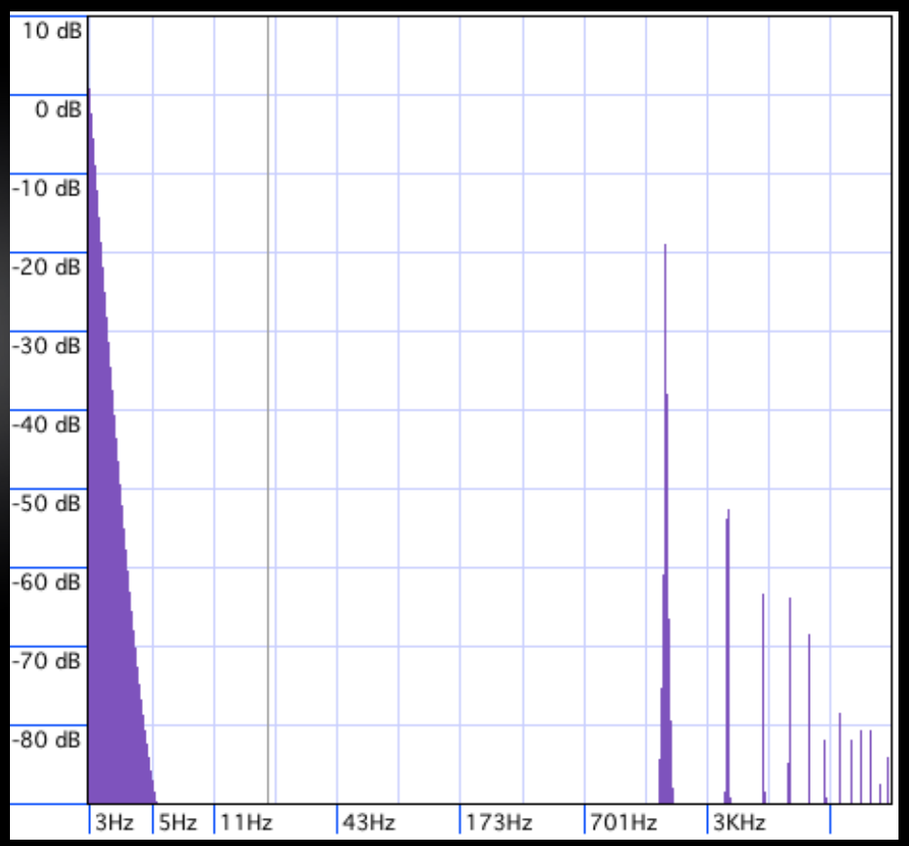


まず皆さんご存知の事かと思いますが基本的な事柄について確認しておきましょう。音にはドレミファソラシドといった音階があり、音楽を演奏する為にはこれらを出し分ける必要があります。音階の違いとは音の周波数の違いです。例えば、サイン波でラの音を出すと、その周波数のピークは★このあたりになるのですが、1オクターブ高いラの音を出すと、その周波数のピークは★このあたりに移動します。

含まれている周波数成分によって音色が変わる

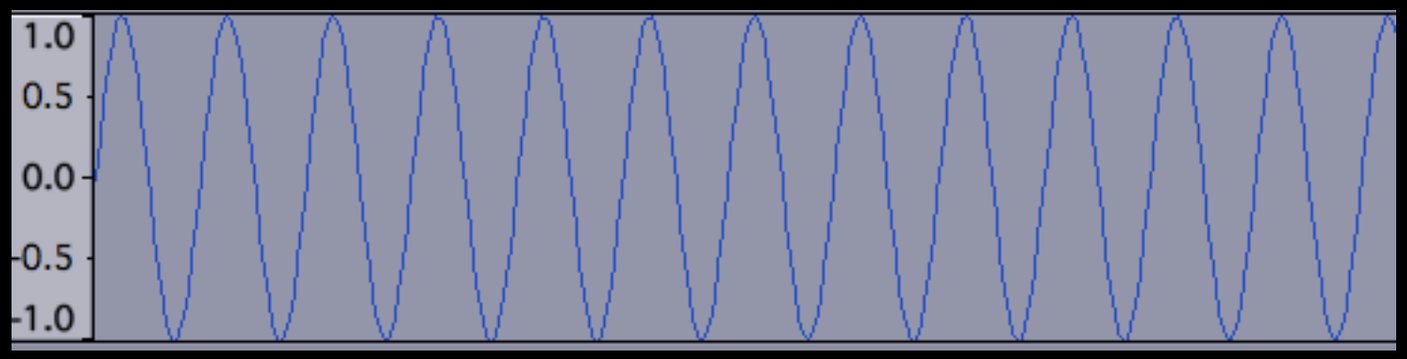


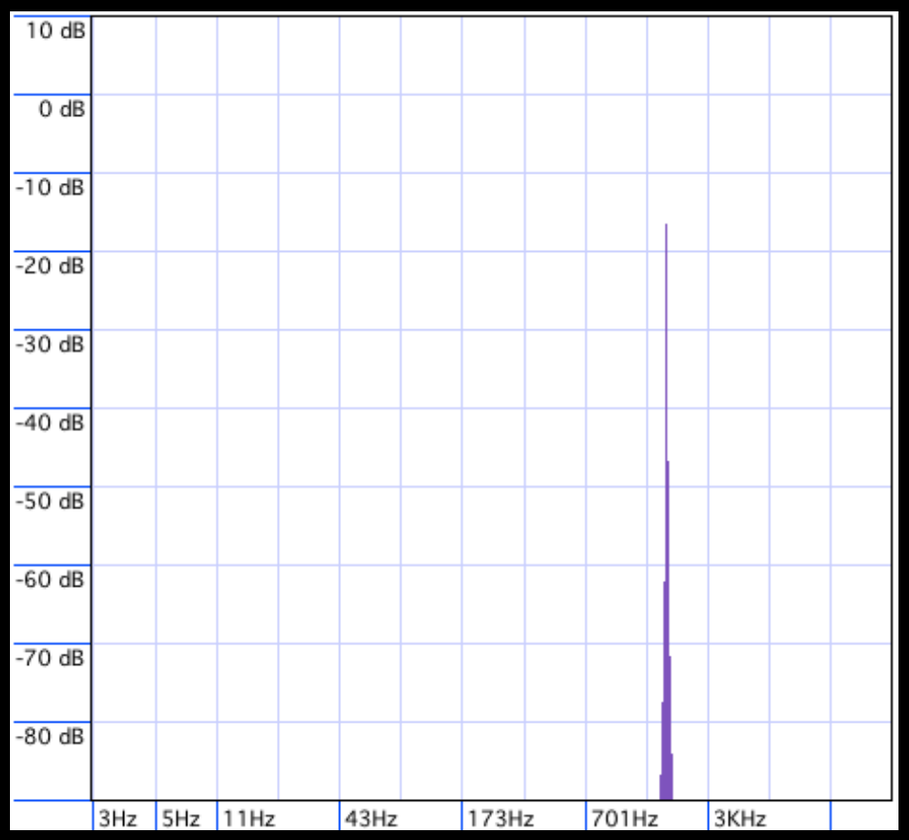


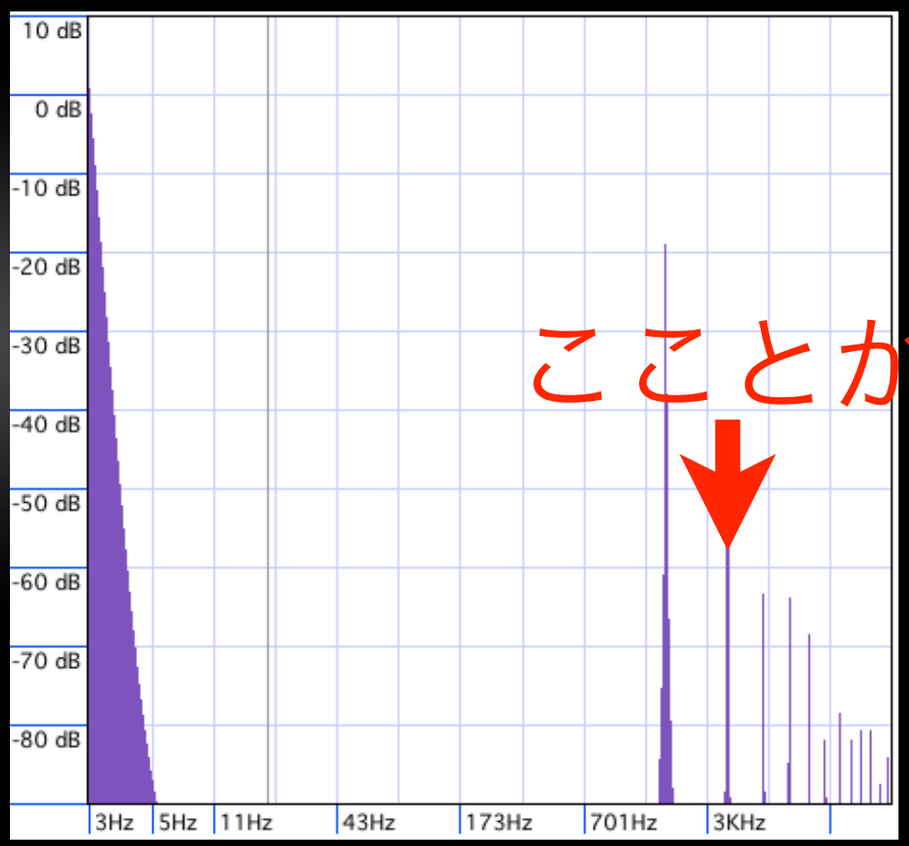


同じ音階でも世の中には色んな音色を奏でる楽器があります。それらは何が違うのかというと、★このように、最も強い周波数成分以外に含まれている周波数成分の分布が異なっています。

含まれている周波数成分によって音色が変わる

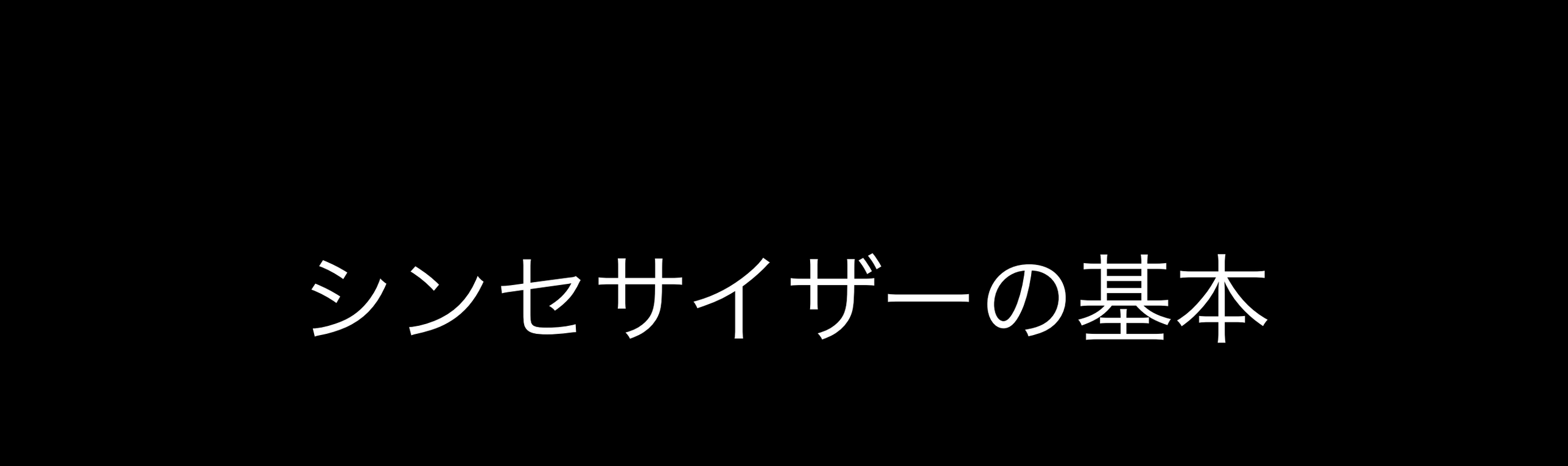






同じ音階でも世の中には色んな音色を奏でる楽器があります。それらは何が違うのかというと、★このように、最も強い周波数成分以外に含まれている周波数成分の分布が異なっています。

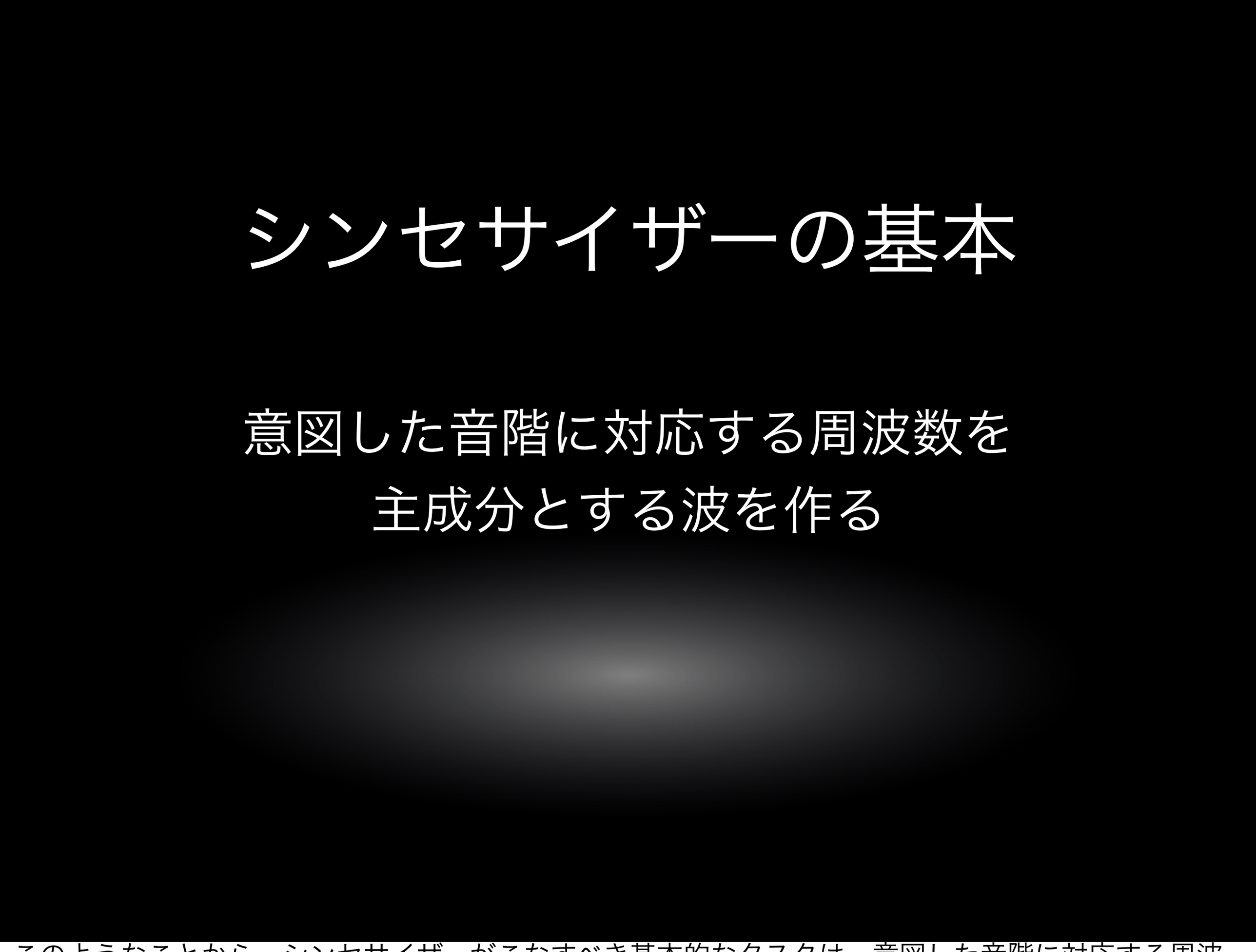
シンセサイザーの基本



このようなことから、シンセサイザーがこなすべき基本的なタスクは、意図した音階に対応する周波数を主成分とする波を生成する事であるということができます。これをPCで実装しても良いのですが、それでは面白みに欠けるため、

シンセサイザーの基本

意図した音階に対応する周波数を
主成分とする波を作る

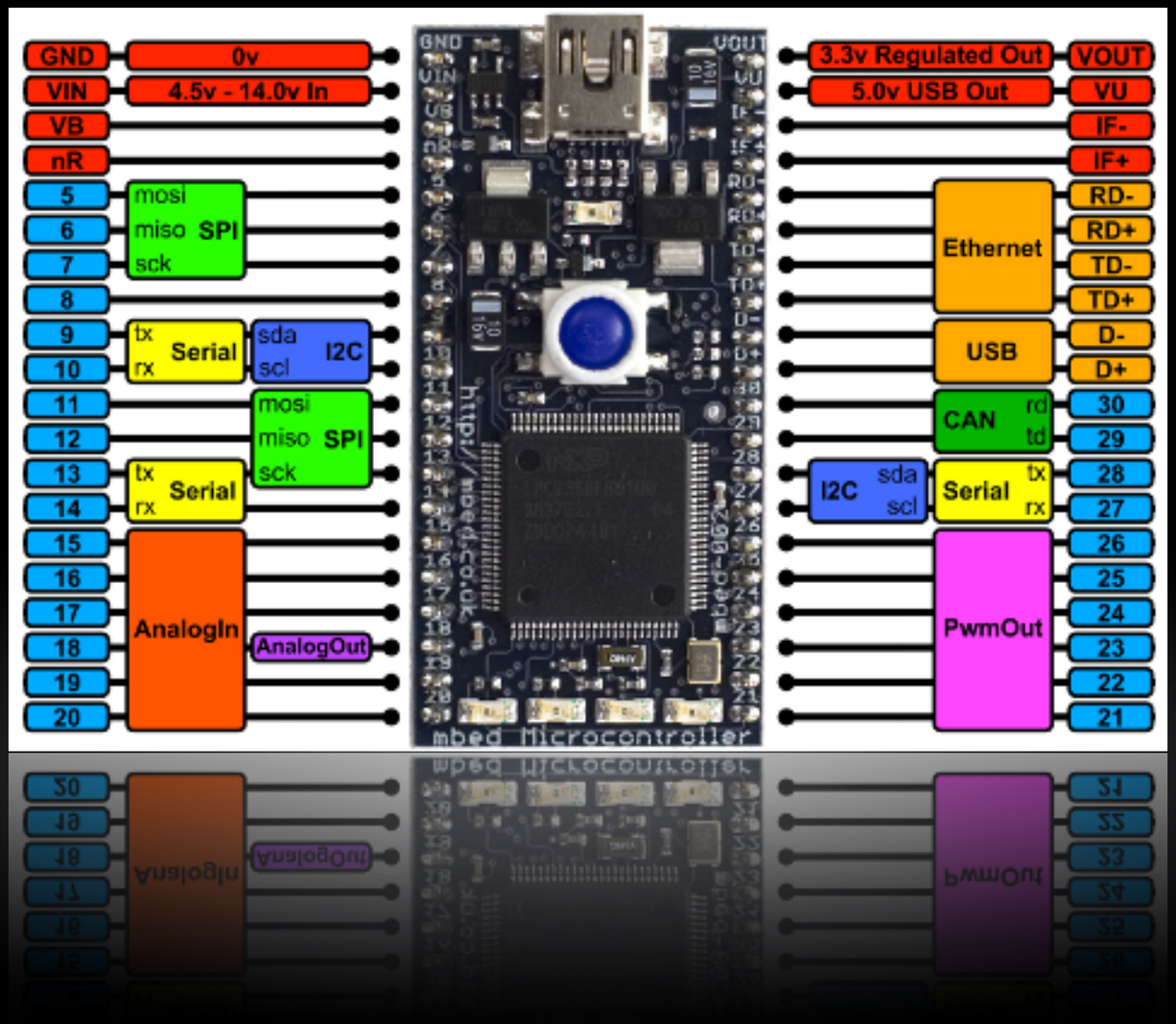


このようなことから、シンセサイザーがこなすべき基本的なタスクは、意図した音階に対応する周波数を主成分とする波を生成する事であるということができます。これをPCで実装しても良いのですが、それでは面白みに欠けるため、

流行のマイコンmbedさん

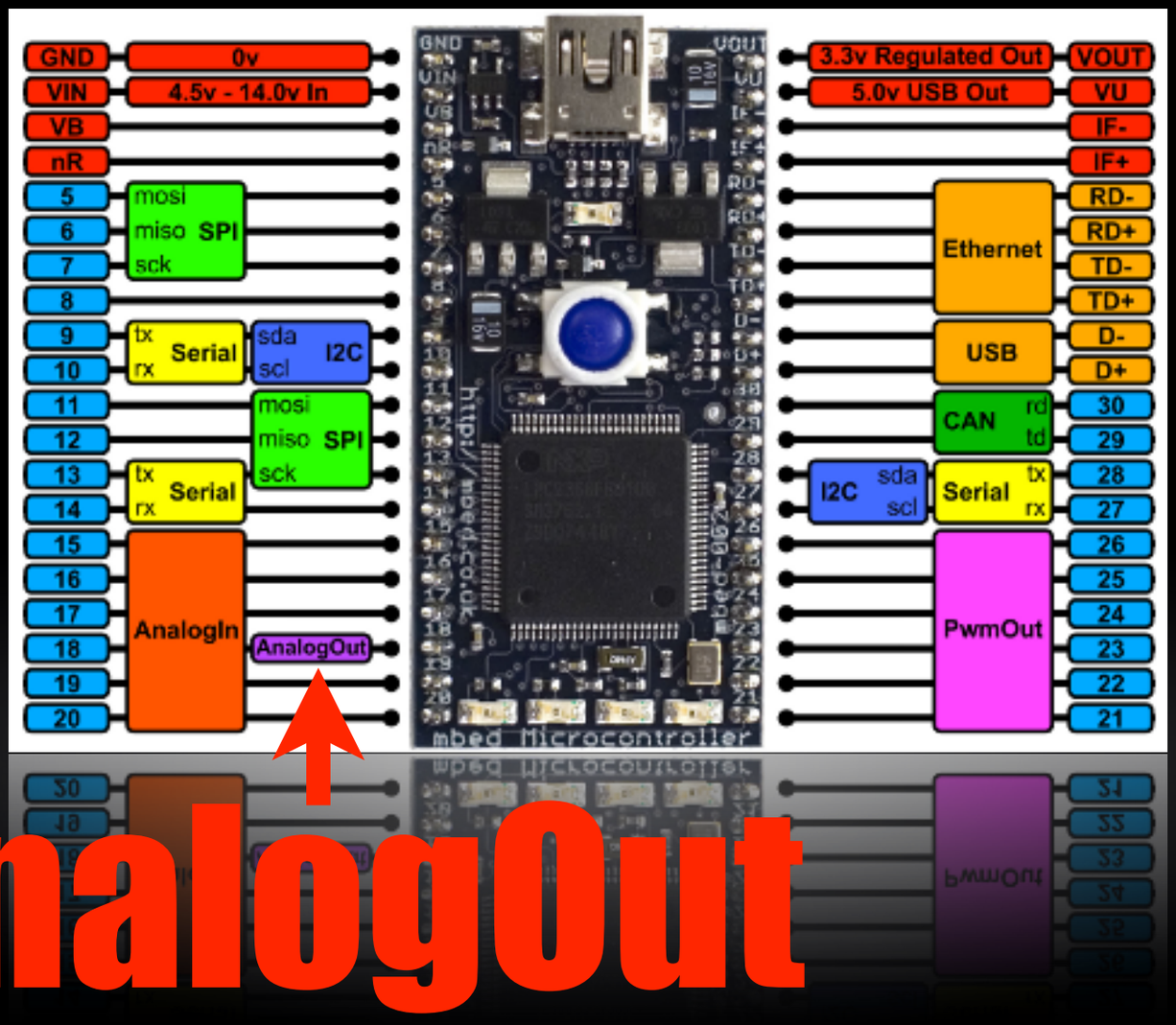
流行のマイコン、mbedさんに組み込む事にしました。★mbedはこのように多彩なI/Oを備えたARMマイコンボードなのですが、★この部分になんとアナログ出力を備えています。これはもうシンセサイザーを作れと言っているようなものですね。

流行のマイコンmbedさん



流行のマイコン、mbedさんに組み込む事にしました。★mbedはこのように多彩なI/Oを備えたARMマイコンボードなのですが、★この部分になんとアナログ出力を備えています。これはもうシンセサイザーを作れと言っているようなものですね。

流行のマイコンmbedさん

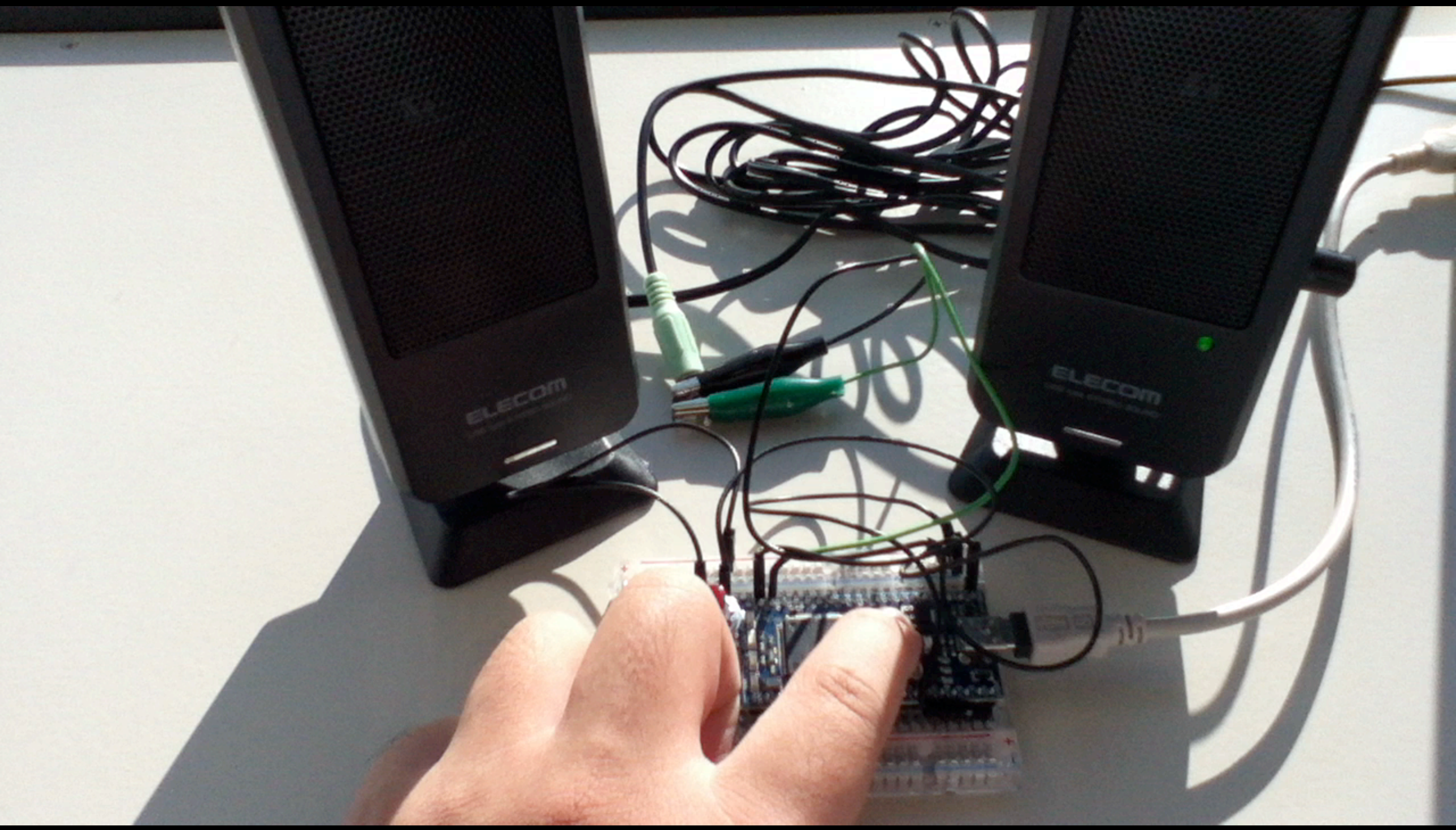


流行のマイコン、mbedさんに組み込む事にしました。★mbedはこのように多彩なI/Oを備えたARMマイコンボードなのですが、★この部分になんとアナログ出力を備えています。これはもうシンセサイザーを作れと言っているようなものですね。

```
#include "mbed.h"
AnalogOut audio_out(p18);
const float freq_table[ 8 ] = {
  523.25113f, 587.32953f, 659.25511f, 698.45646f,
  783.99087f, 880.00000f, 987.76660f, 1046.5022f,
};
int main() {
  while(1) {
    for( int note = 0; note != 8; ++note )
      for( float time = 0.0f; time < 1.0f; time += 1.0f/16000.0f ) {
        Timer used_time;
        used_time.start();
        audio_out =
          sinf( time * freq_table[ note ] * 3.141592f * 2.0f ) *
          0.5f + 0.5f;
        used_time.stop();
        wait(1.0f/16000.0f-used_time.read());
    }
  }
}
```

```
}
  }
    }
```

それではまず簡単な音を鳴らすコードを書いてみましょう。緑で示した部分でサイン波を作って、アナログ出力にそのまま書き込んでいます。これをmbedで動かすとこうなります。

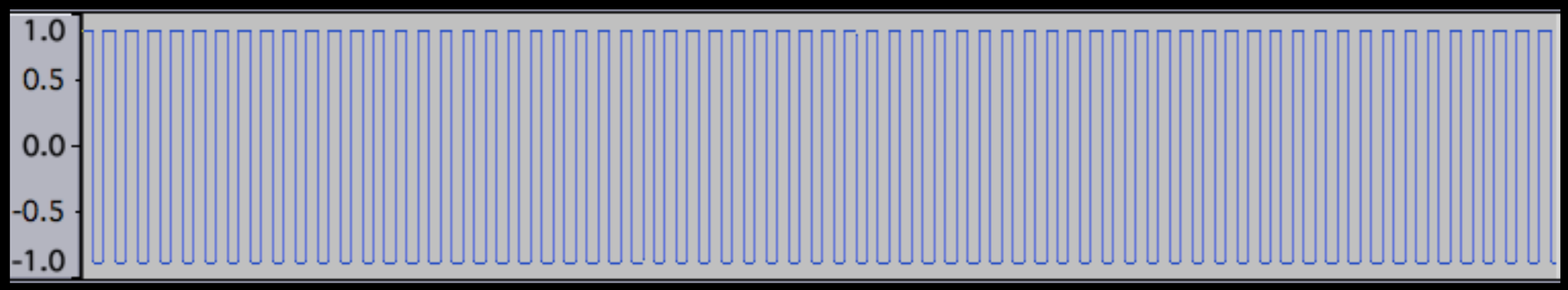

ELECOM
ELECOM

もっと複雑な音を出したい

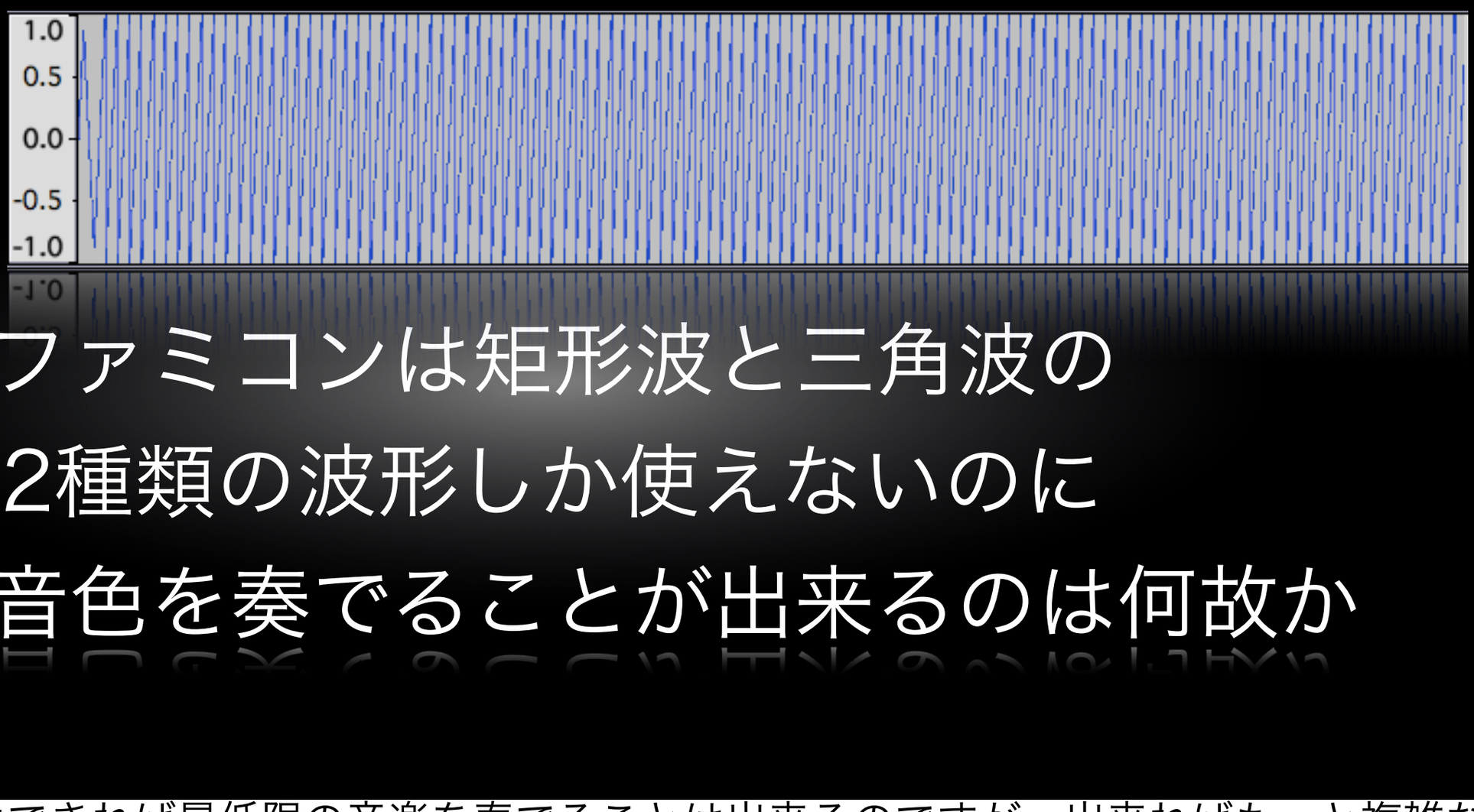
さて、サイン波が出力できれば最低限の音楽を奏でることは出来るのですが、出来ればもっと複雑な音色を奏でたいですよね。★ファミリーコンピュータの音はおそらく世界で最も有名な電子楽器の音ではないかと思うのですが、あの音源には矩形波と三角波の2種類のシンプルな波形しか用意されていません。にもかかわらず、ファミリーコンピュータが様々な表情を持った音色を奏でることが出来たのは何故でしょうか。

もっと複雑な音を出したい

矩形波



三角波



ファミコンは矩形波と三角波の
2種類の波形しか使えないのに
多彩な音色を奏でることが出来るのは何故か

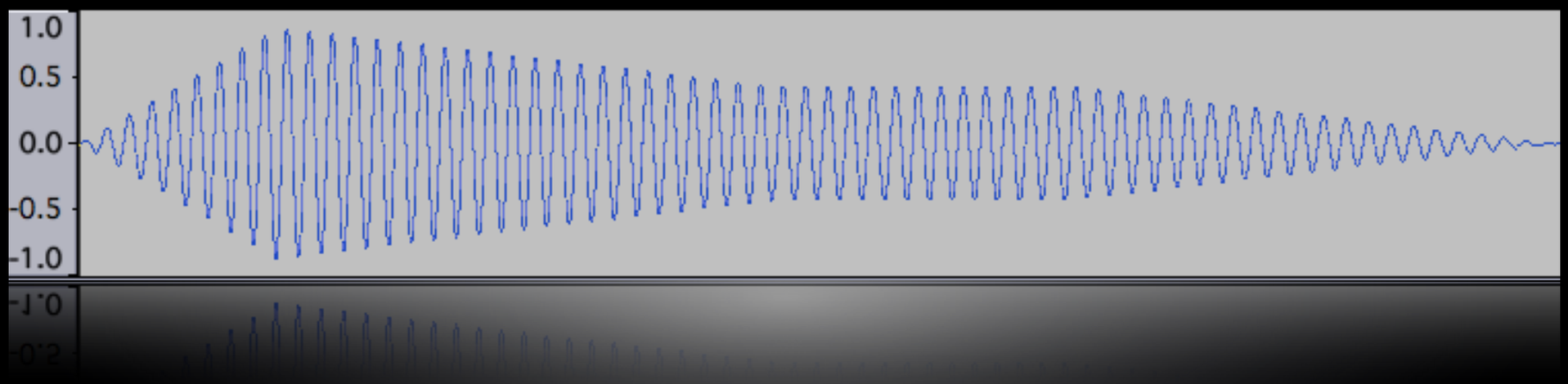
さて、サイン波が出力できれば最低限の音楽を奏でることは出来るのですが、出来ればもっと複雑な音色を奏でたいですよね。★ファミリーコンピュータの音はおそらく世界で最も有名な電子楽器の音ではないかと思うのですが、あの音源には矩形波と三角波の2種類のシンプルな波形しか用意されていません。にもかかわらず、ファミリーコンピュータが様々な表情を持った音色を奏でることが出来たのは何故でしょうか。

もっと複雑な音を出したい

エンベロープ

ファミリーコンピュータの音色に豊かな表情を与えているのはエンベロープと呼ばれる仕組みです。★エンベロープは、音を鳴らしはじめてからの音量を経過時間に応じて変化させることで、アコースティックな楽器の挙動に似た音色を作り出す仕組みです。

もっと複雑な音を出したい

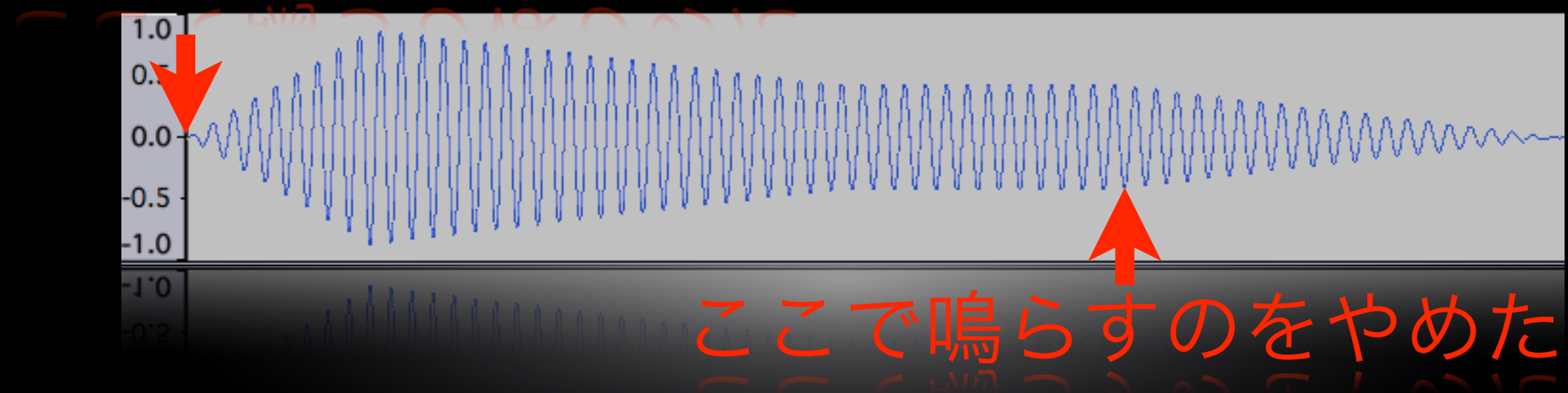


エンベロープ

ファミリーコンピュータの音色に豊かな表情を与えているのはエンベロープと呼ばれる仕組みです。★エンベロープは、音を鳴らしはじめてからの音量を経過時間に応じて変化させることで、アコースティックな楽器の挙動に似た音色を作り出す仕組みです。

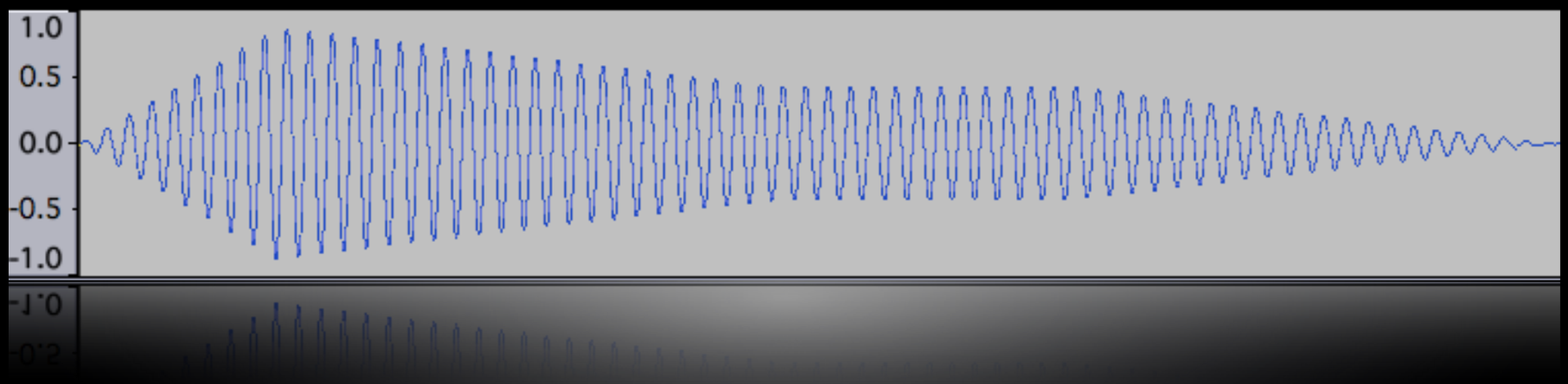
もっと複雑な音を出したい

音量を経過時間に応じて変化させる



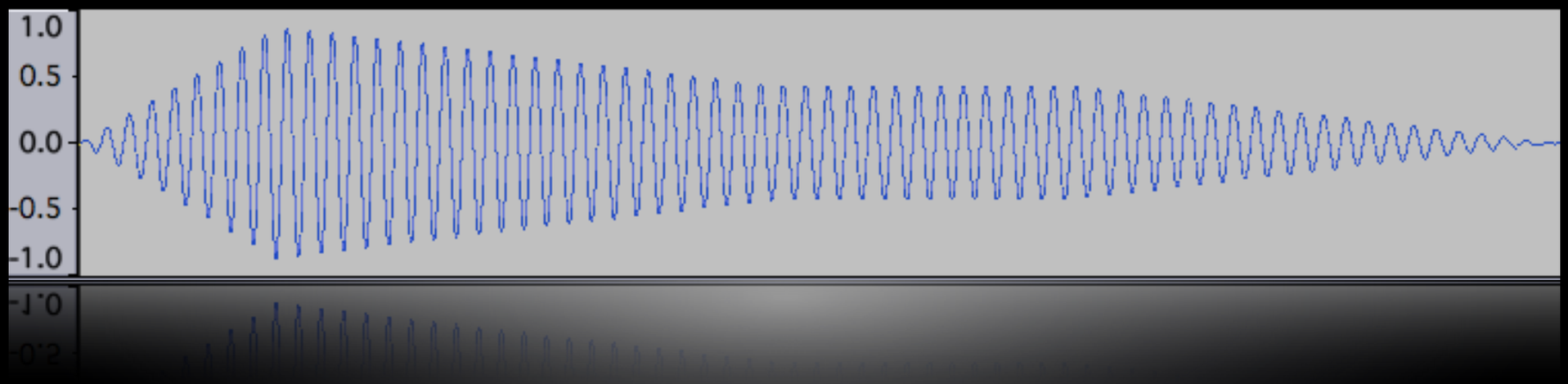
エンベロープ

ファミリーコンピュータの音色に豊かな表情を与えているのはエンベロープと呼ばれる仕組みです。★エンベロープは、音を鳴らしはじめてからの音量を経過時間に応じて変化させることで、アコースティックな楽器の挙動に似た音色を作り出す仕組みです。



★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

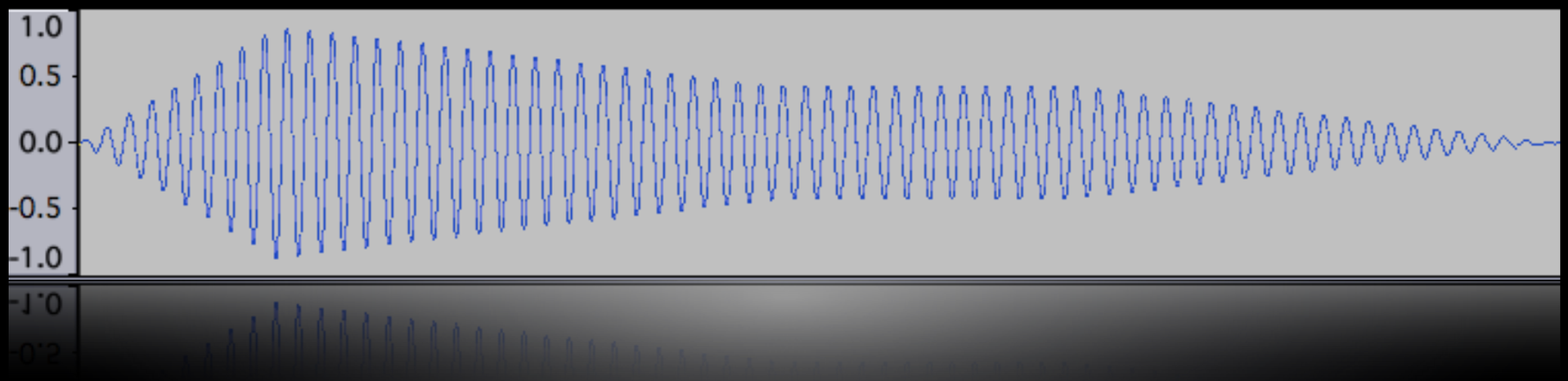
ピアノ



★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

ピアノ

鳴りはじめの
音量が最も大きい

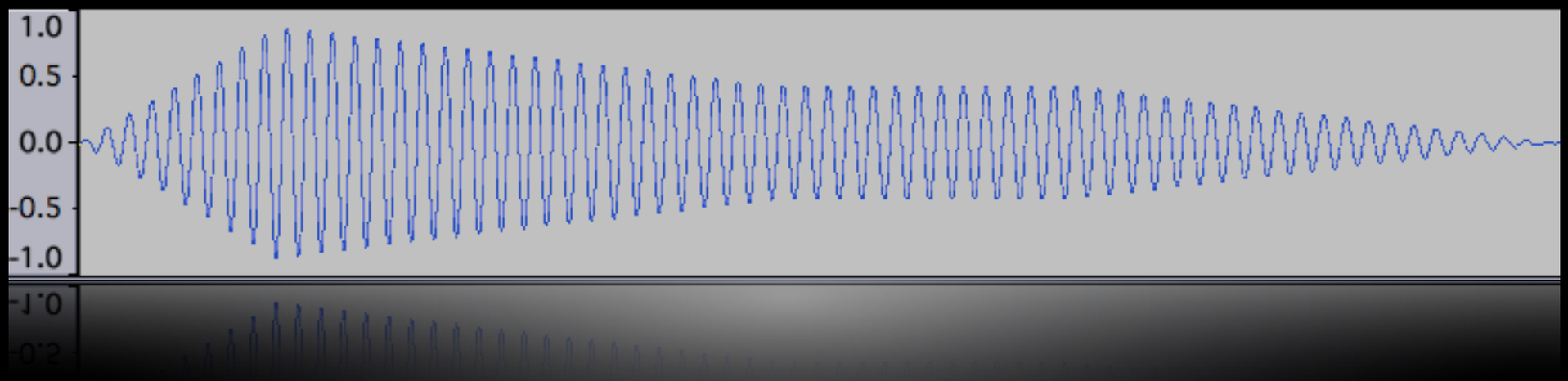


★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

ピアノ

鳴りはじめの
音量が最も大きい

バイオリン



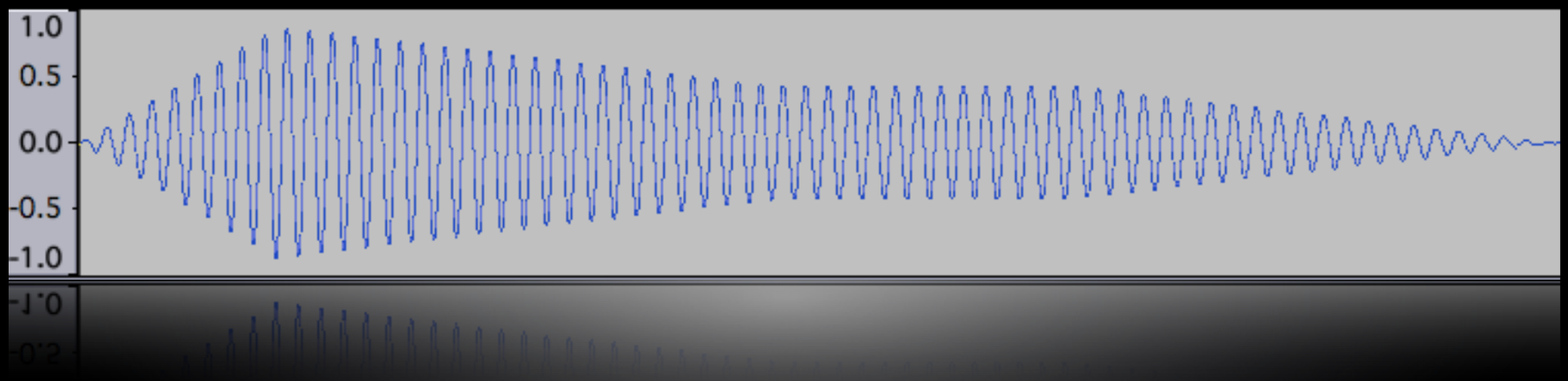
★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

ピアノ

鳴りはじめの
音量が最も大きい

バイオリン

鳴りはじめてから
徐々に音量が大きくなる



★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

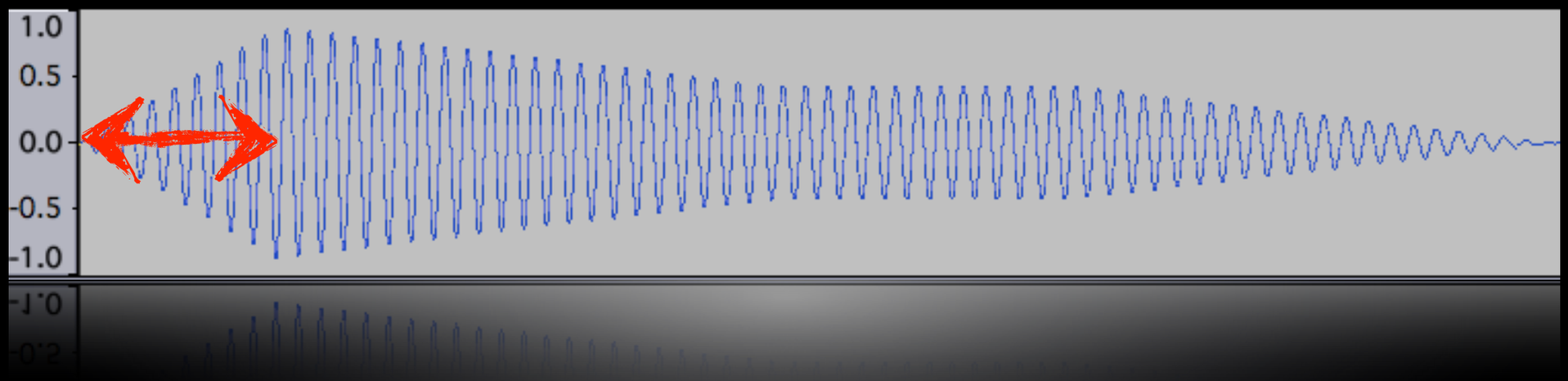
ピアノ

鳴りはじめの
音量が最も大きい

バイオリン

鳴りはじめてから
徐々に音量が大きくなる

この部分の長さを変えることで対応



★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。

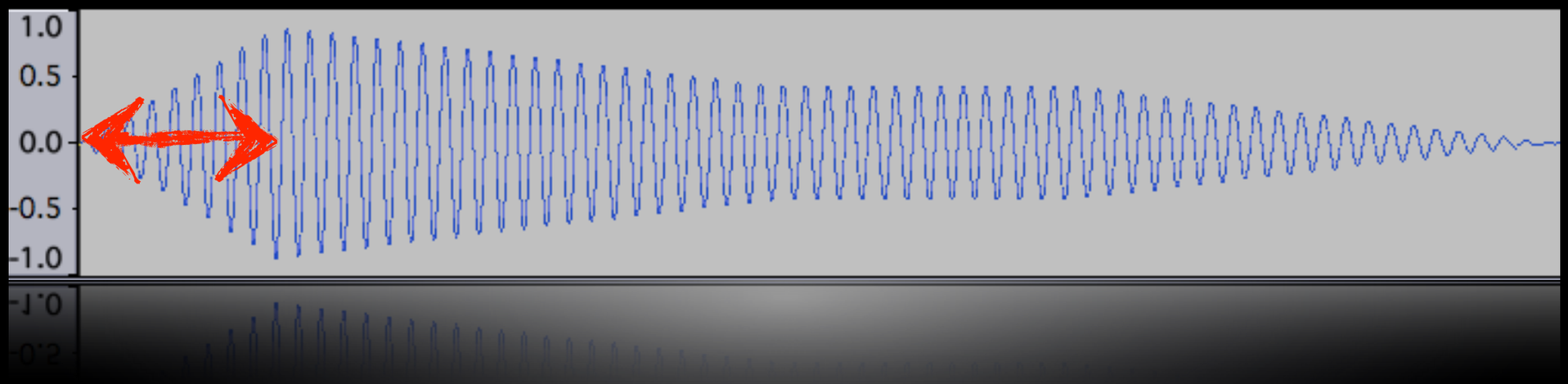
ピアノ

鳴りはじめの
音量が最も大きい

バイオリン

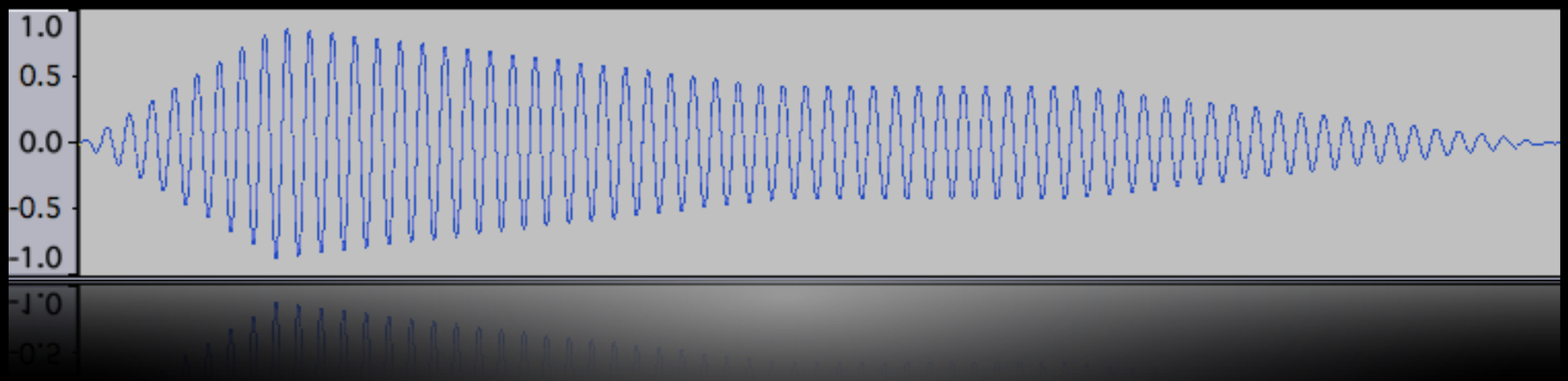
鳴りはじめてから
徐々に音量が大きくなる

この部分の長さを変えることで対応



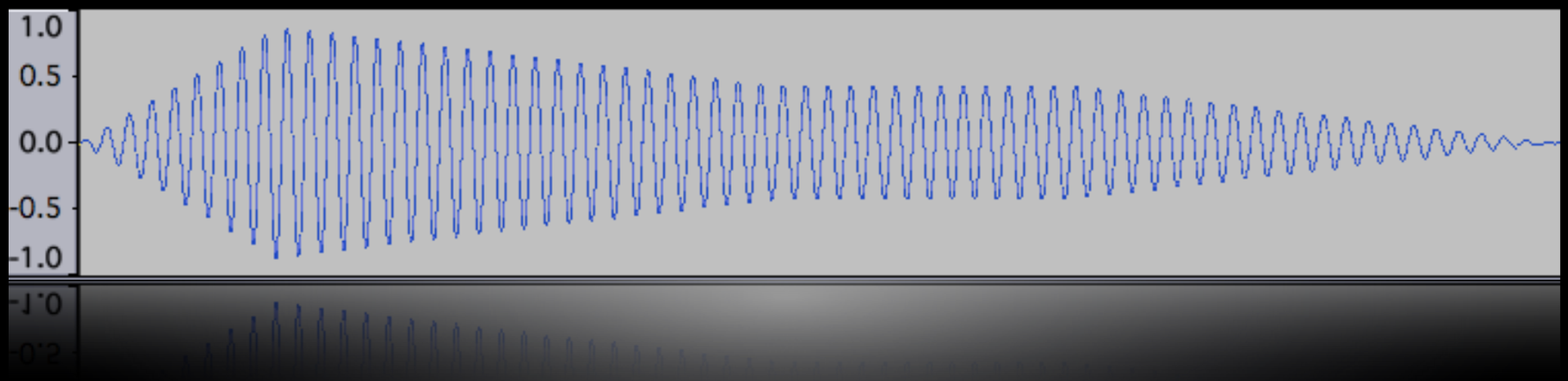
アタック

★ピアノを思い浮かべてください。★ピアノは鍵盤を叩いた瞬間に最も大きな音が出ます。★一方バイオリンはどうでしょうか。★バイオリンは鳴りはじめてから音量が最大に達するまでに少し時間を要します。★音を鳴らしはじめてから最大音量に達するまでの時間を調節できるようにすることで、これらの楽器の違いを表現することが出来ます。★この部分の時間の長さをアタックと呼びます。



★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

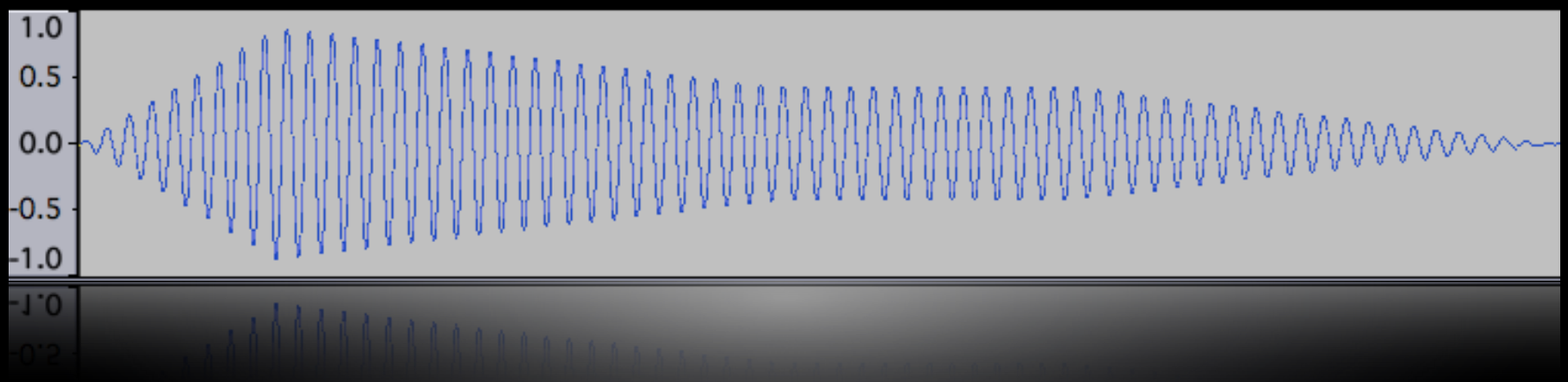
シロフォン



★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

シロフォン

鳴らしはじめて最大音量に達した後
すぐに音量が下がる

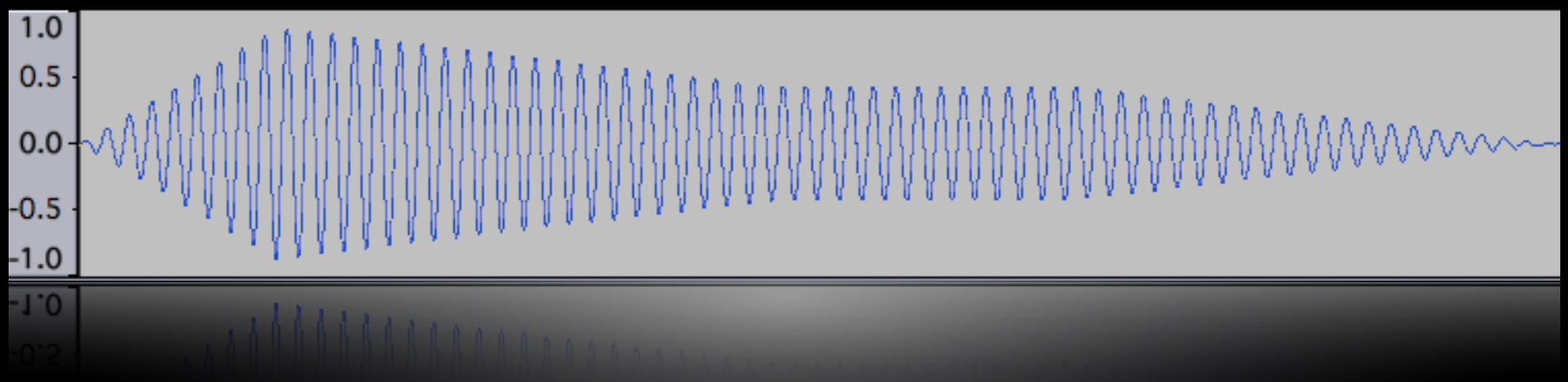


★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

シロフォン

鳴らしはじめて最大音量に達した後
すぐに音量が下がる

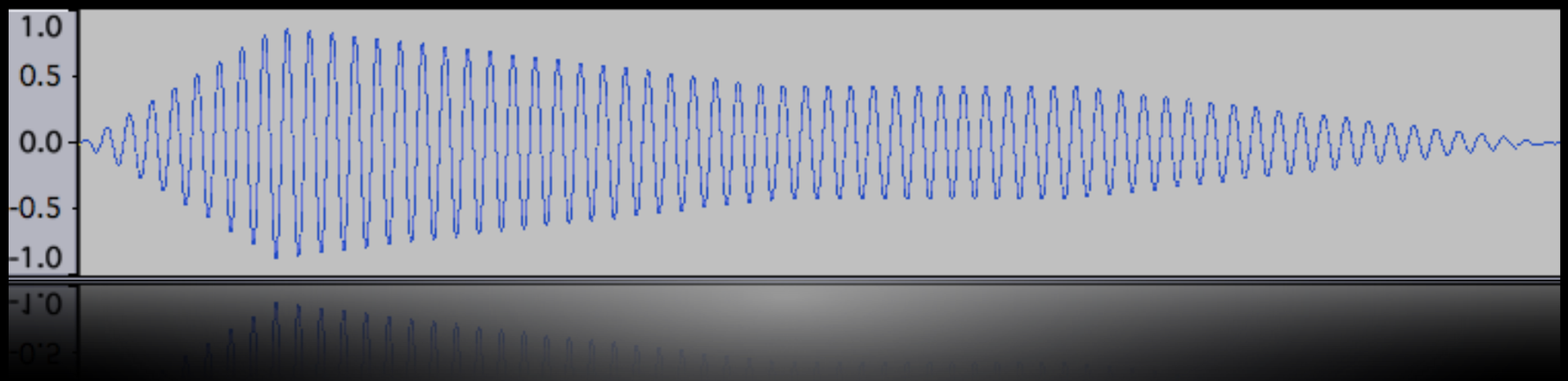
ギター



★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

シロフォン　鳴らしはじめて最大音量に達した後すぐに音量が下がる

ギター　鳴らしはじめて最大音量に達した後ゆっくり音量が下がる

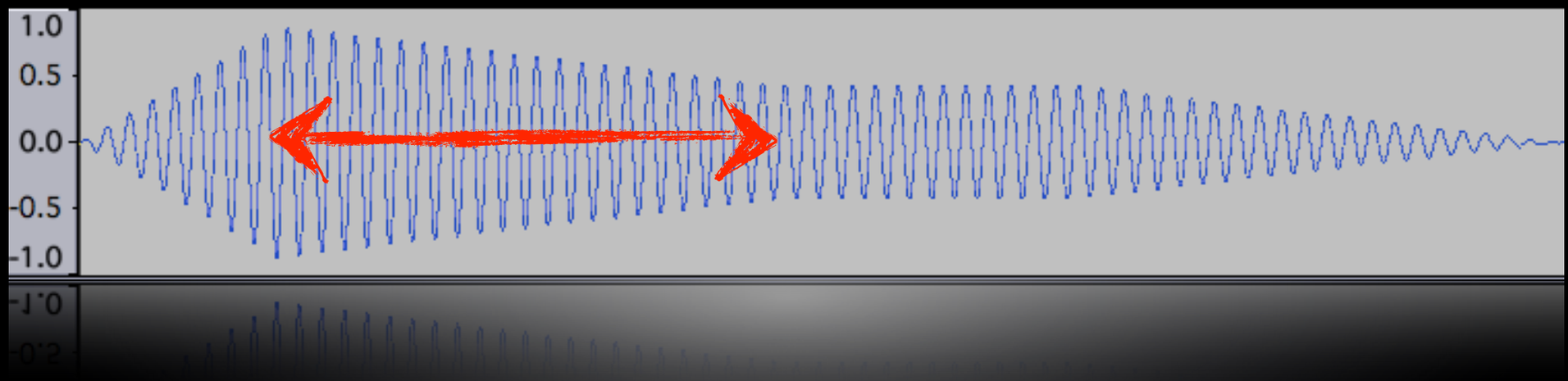


★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

シロフォン　鳴らしはじめて最大音量に達した後すぐに音量が下がる

ギター　鳴らしはじめて最大音量に達した後ゆっくり音量が下がる

この部分の長さを変えることで対応

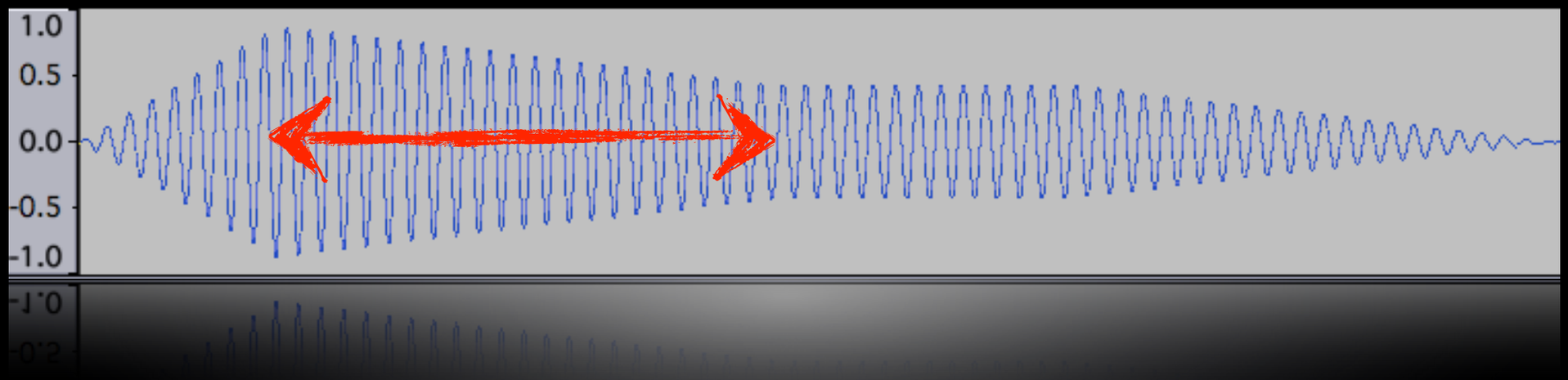


★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。

シロフォン　鳴らしはじめて最大音量に達した後すぐに音量が下がる

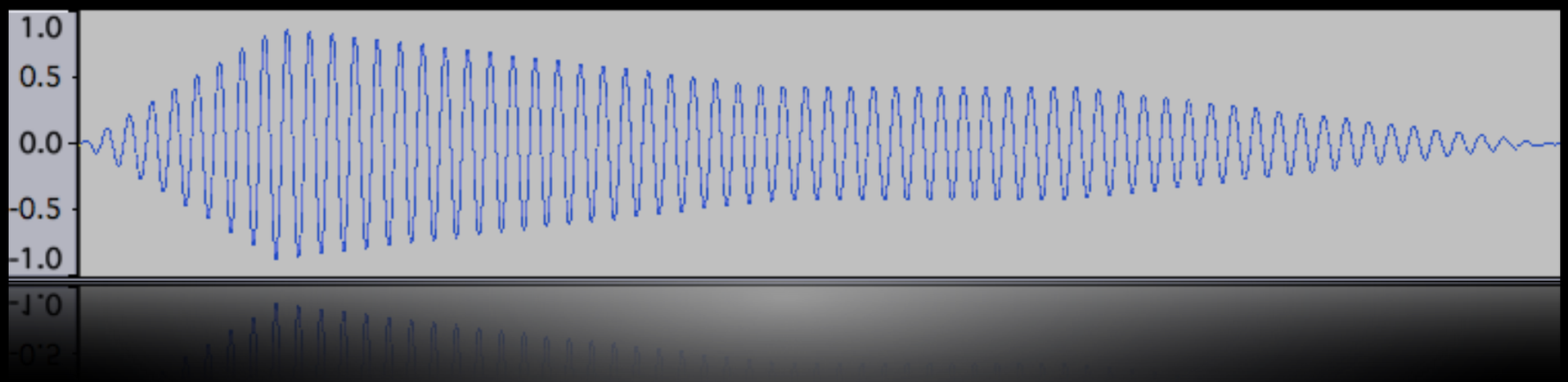
ギター　鳴らしはじめて最大音量に達した後ゆっくり音量が下がる

この部分の長さを変えることで対応



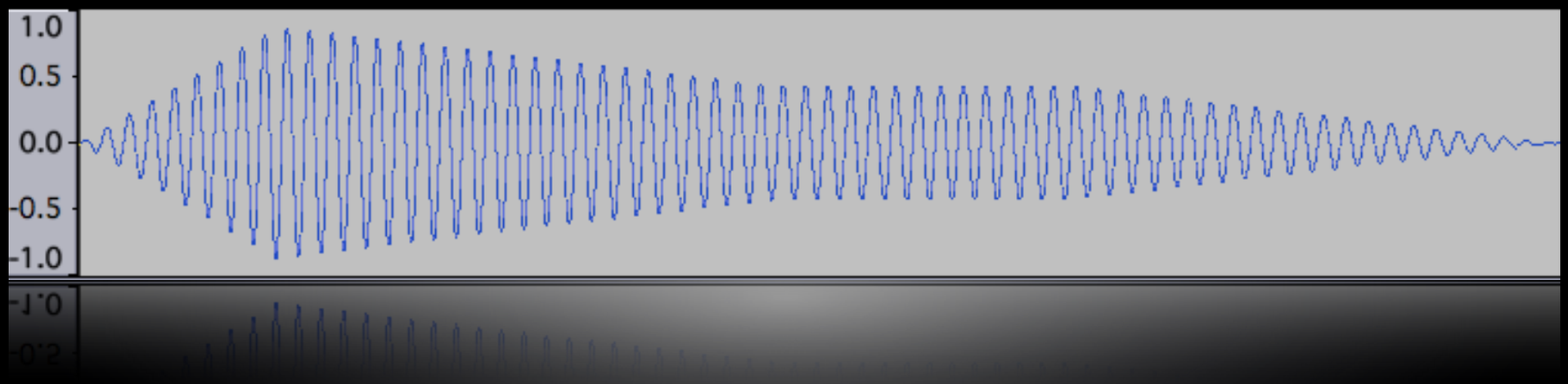
ディケイ

★シロフォンを想像してください。★シロフォンは叩いて最大音量に達した後、すぐに音量が下がります。★一方ギターはどうでしょうか。★ギターは一度弦を弾くと、最大音量に達した後暫く鳴り続けます。★最大音量に達してから音量が下がるのに要する時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の時間の長さをディケイと呼びます。



★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

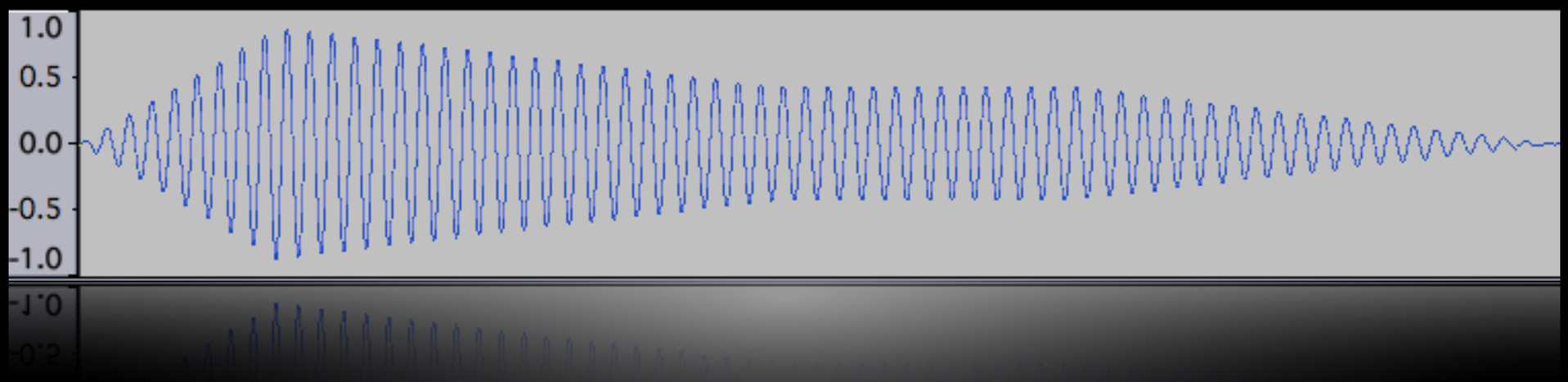
ピアノ



★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

ピアノ

鍵盤を押している時間に関わらず
ある程度の時間で音が消える

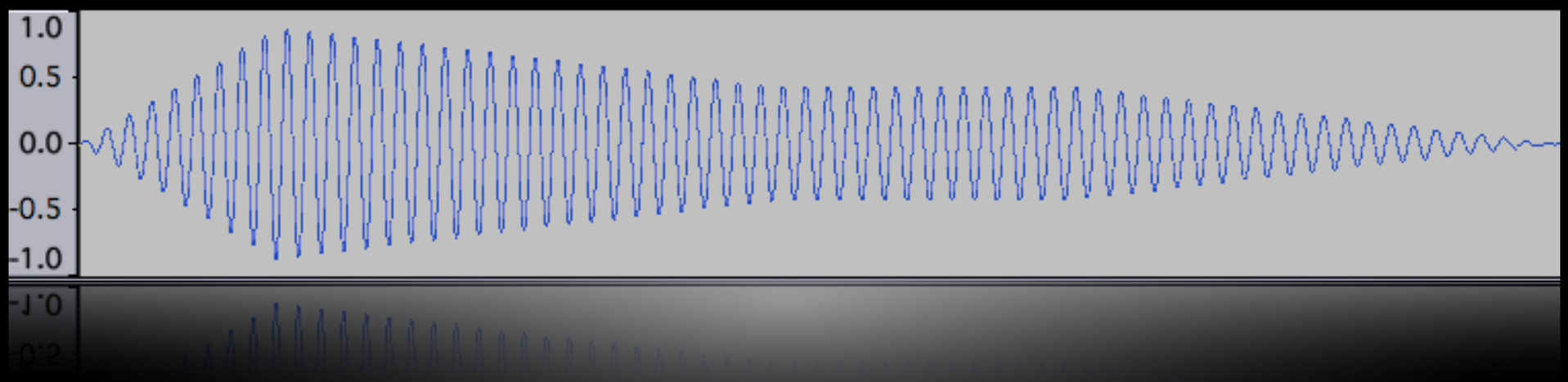


★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

ピアノ

鍵盤を押している時間に関わらず
ある程度の時間で音が消える

オルガン



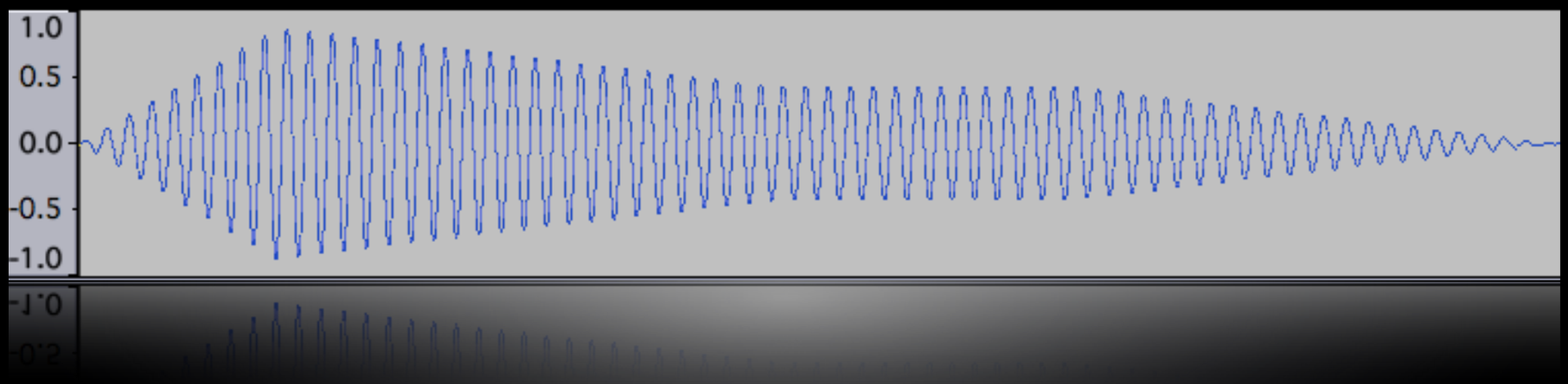
★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

ピアノ

鍵盤を押している時間に関わらず
ある程度の時間で音が消える

オルガン

鍵盤を押している限り際限なく
音が鳴り続ける

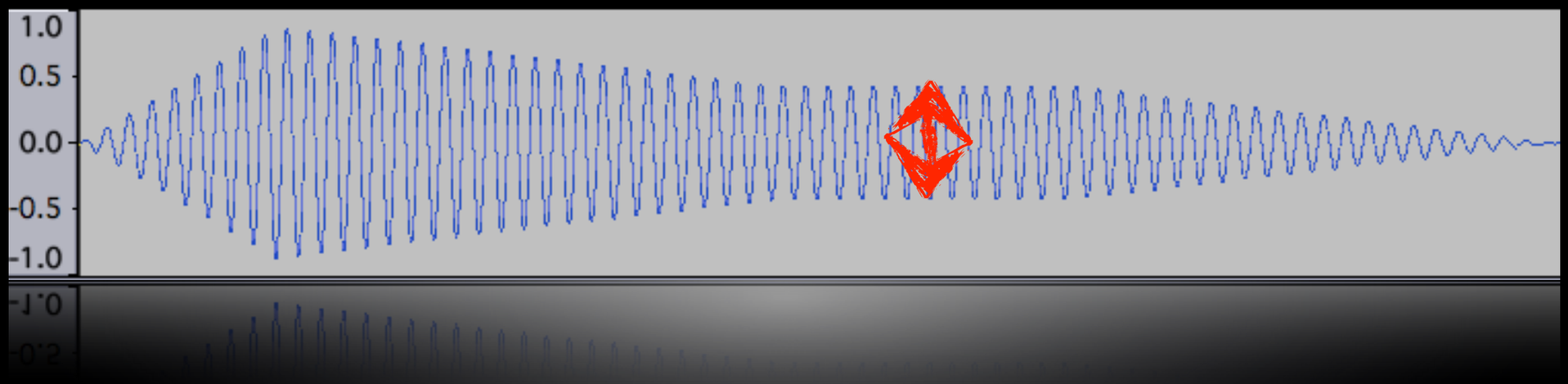


★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

ピアノ　鍵盤を押している時間に関わらず
ある程度の時間で音が消える

オルガン　鍵盤を押している限り際限なく
音が鳴り続ける

この部分の大きさを変えることで対応



★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。

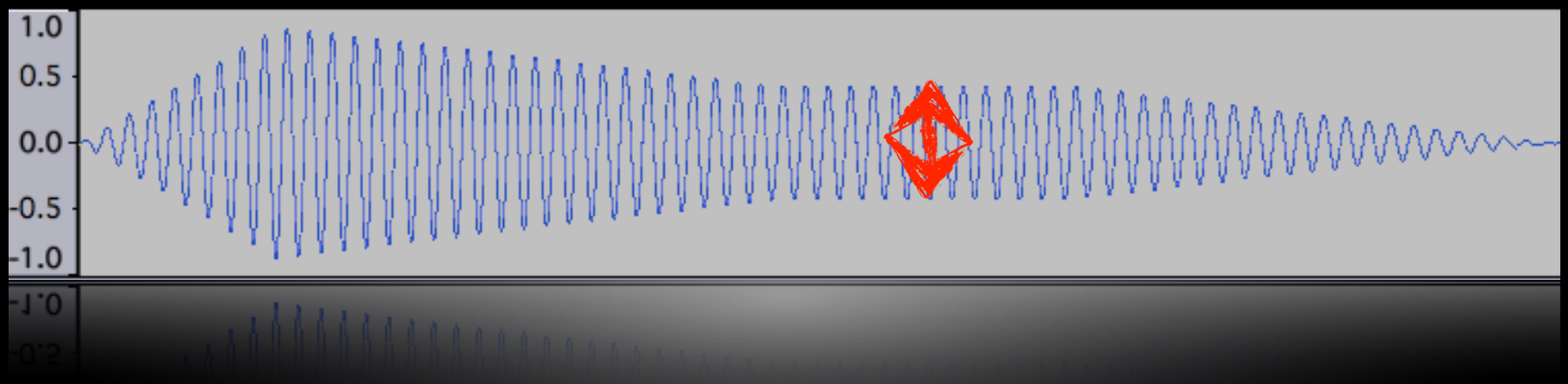
ピアノ

鍵盤を押している時間に関わらず
ある程度の時間で音が消える

オルガン

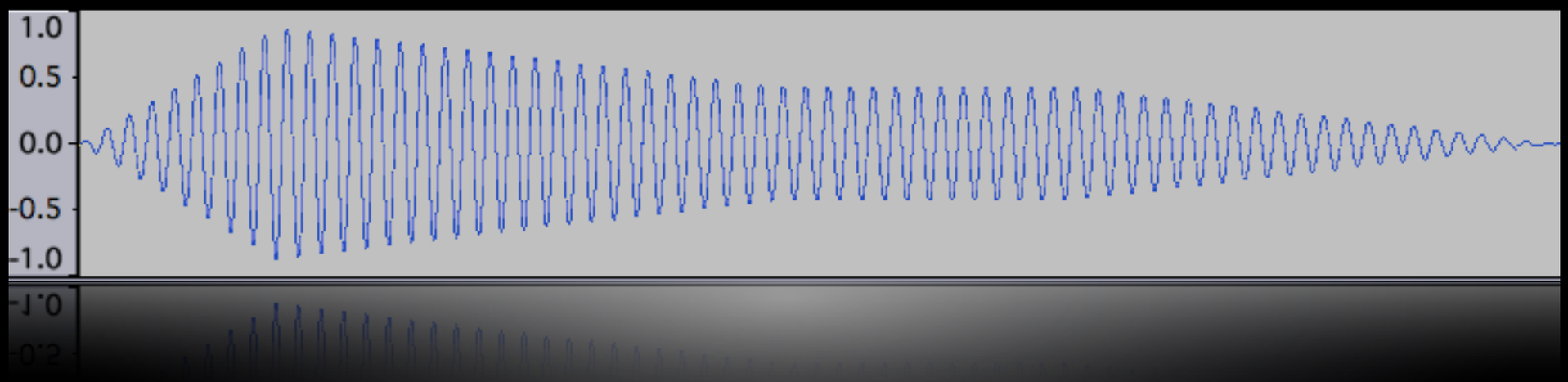
鍵盤を押している限り際限なく
音が鳴り続ける

この部分の大きさを変えることで対応



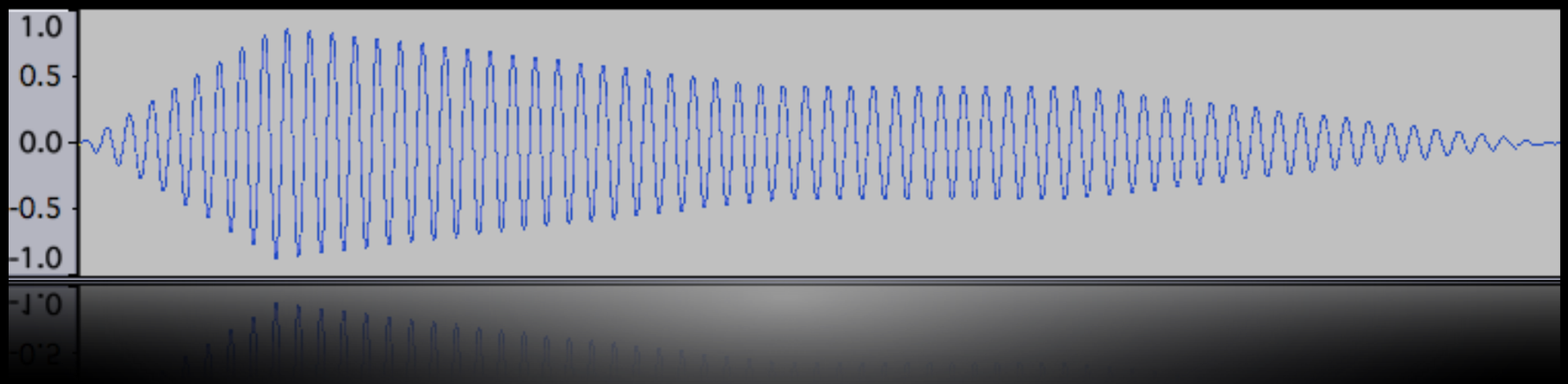
サステイン

★もう一度ピアノを思い浮かべてください。★ピアノはどんなに強く鍵盤を叩いても、ある程度の時間が経過すると音は消えてしまいます。★一方オルガンはどうでしょうか。★オルガンは鍵盤を押している限り、音が勝手に消えてしまうことはありません。★ディケイによって音量がどこまで下がるかを調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の音量をサステインと呼びます。



★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

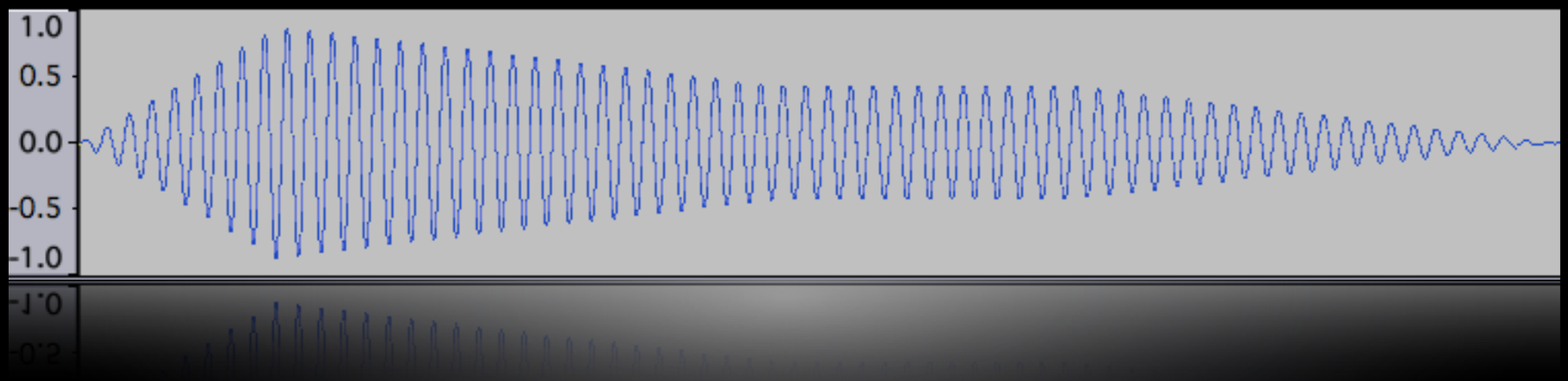
オルガン



★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

オルガン

弾くのをやめると
比較的短時間で音が消える

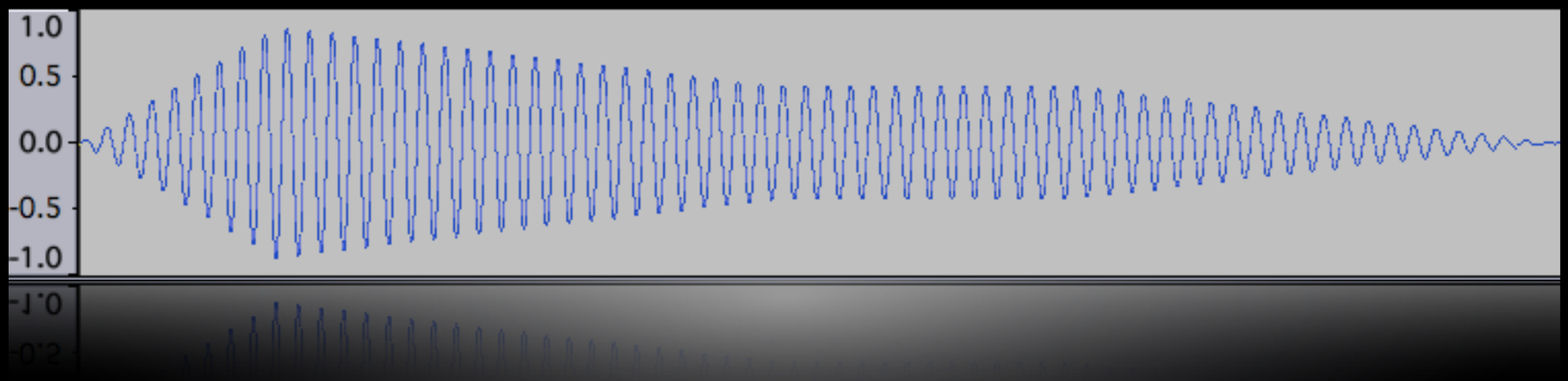


★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

オルガン

弾くのをやめると
比較的短時間で音が消える

スチールドラム



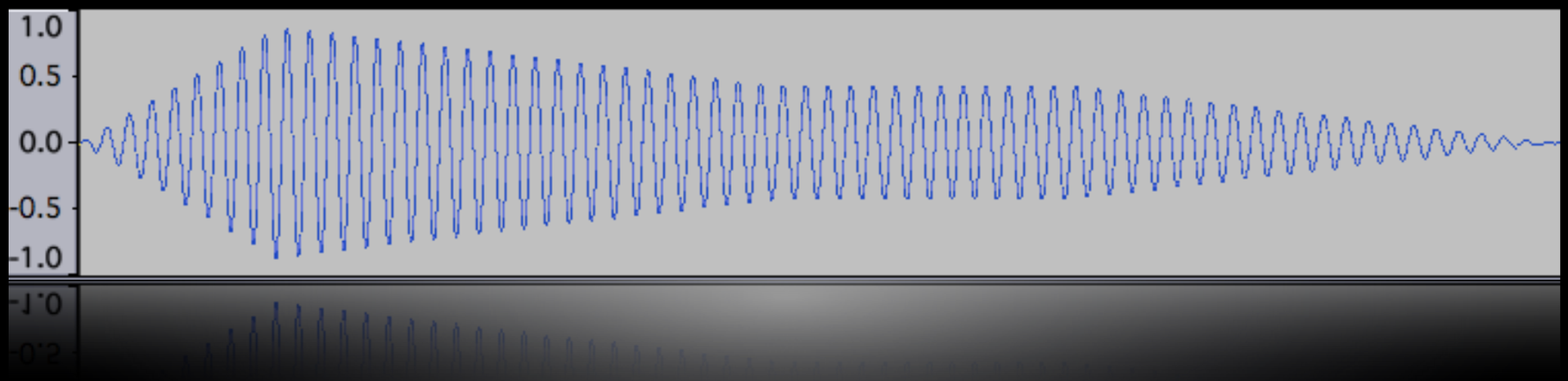
★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

オルガン

スチールドラム

弾くのをやめると
比較的短時間で音が消える

弾くのをやめても音が
消えるまでにある程度の時間を要す



★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

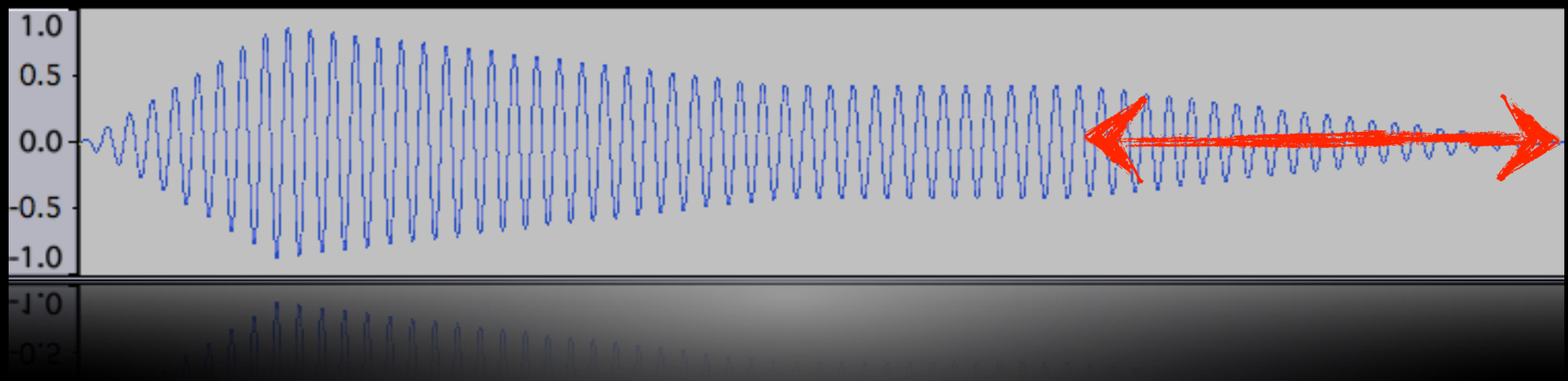
オルガン

スチールドラム

弾くのをやめると
比較的短時間で音が消える

弾くのをやめても音が
消えるまでにある程度の時間を要す

この部分の傾きを変えることで対応



★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。

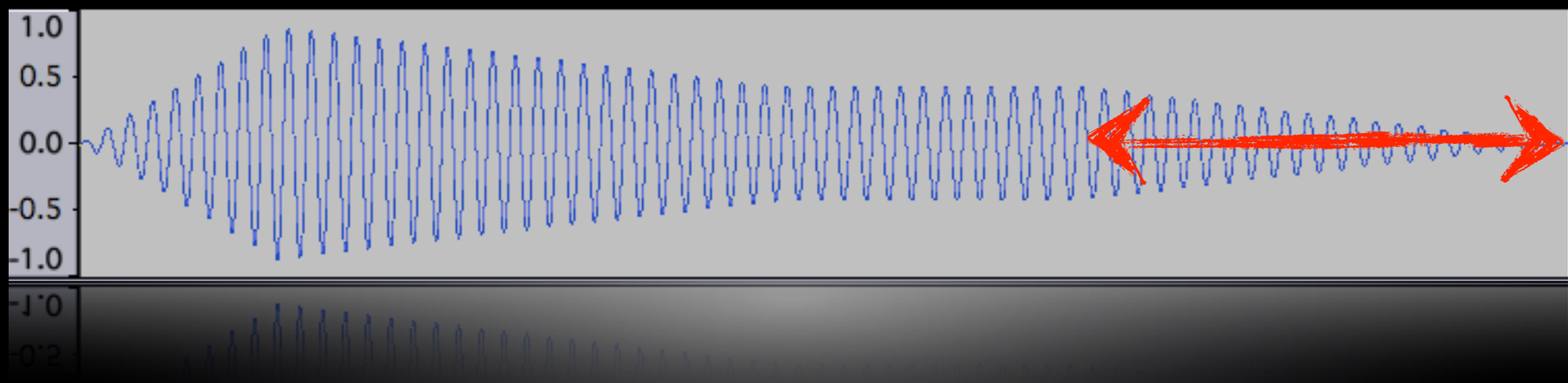
オルガン

スチールドラム

弾くのをやめると
比較的短時間で音が消える

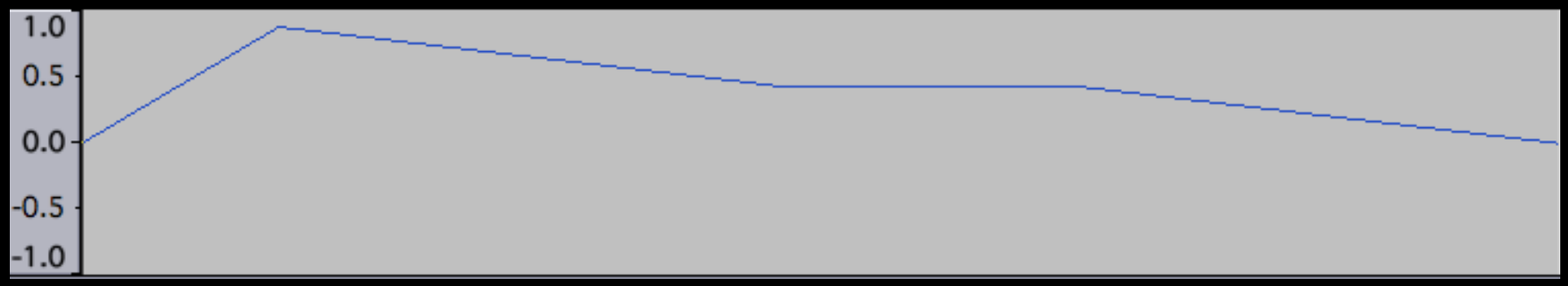
弾くのをやめても音が
消えるまでにある程度の時間を要す

この部分の傾きを変えることで対応



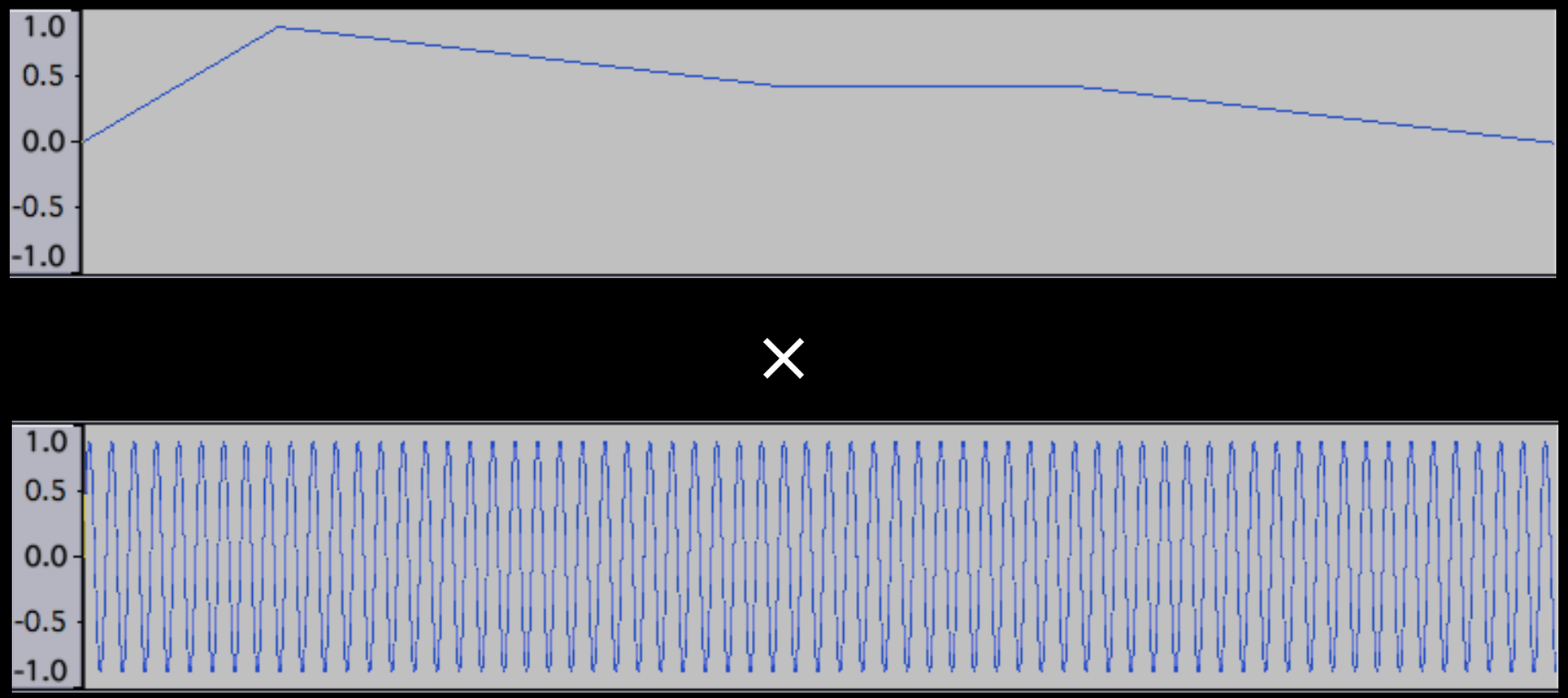
リリース

★もう一度オルガンを思い浮かべてください。★オルガンは鍵盤から指を離すと比較的短時間で音が消えてしまいます。★一方スチールドラムはどうでしょうか。★スチールドラムは叩くのをやめても、暫くの間音が響き続けます。★弾くのをやめてから音が完全に消えるまでの時間を調節できるようにすることで、これらの楽器の違いを表現することができます。★この部分の傾きをリリースと呼びます。



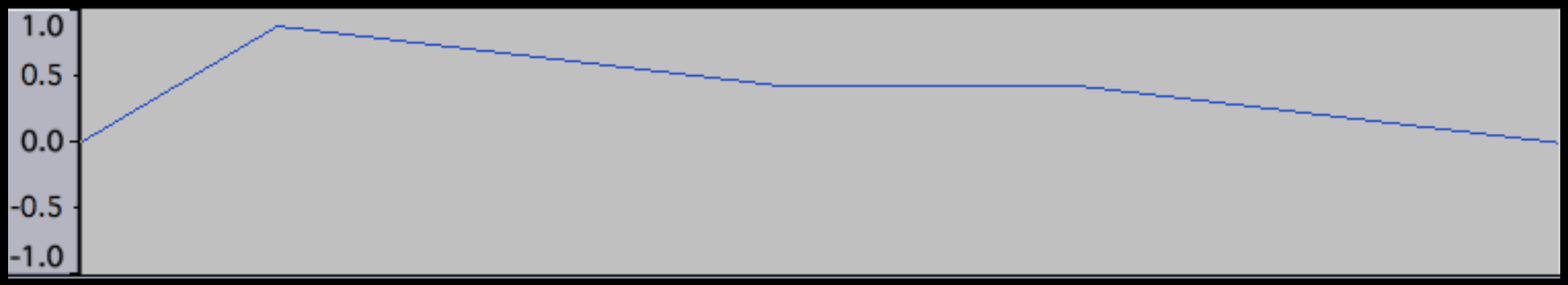
エンベロープ

アタック、ディケイ、サステイン、リリースの値からエンベロープの音量を決定し、★基本となる波との積をとることで、★このような波を得ることができます。シンセサイザーによってはこれら4つのパラメータ以外にもより柔軟なエンベロープを作る為にいくつかのパラメータを持っていることもあります。

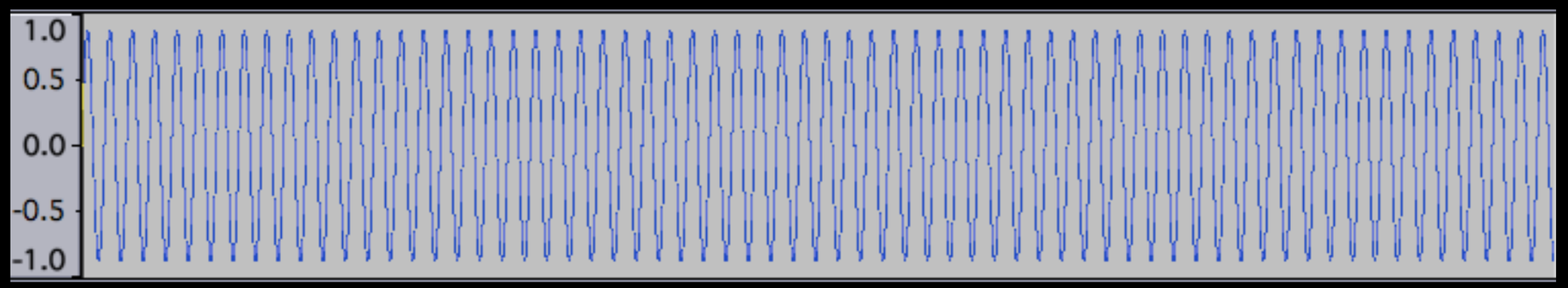


エンベロープ

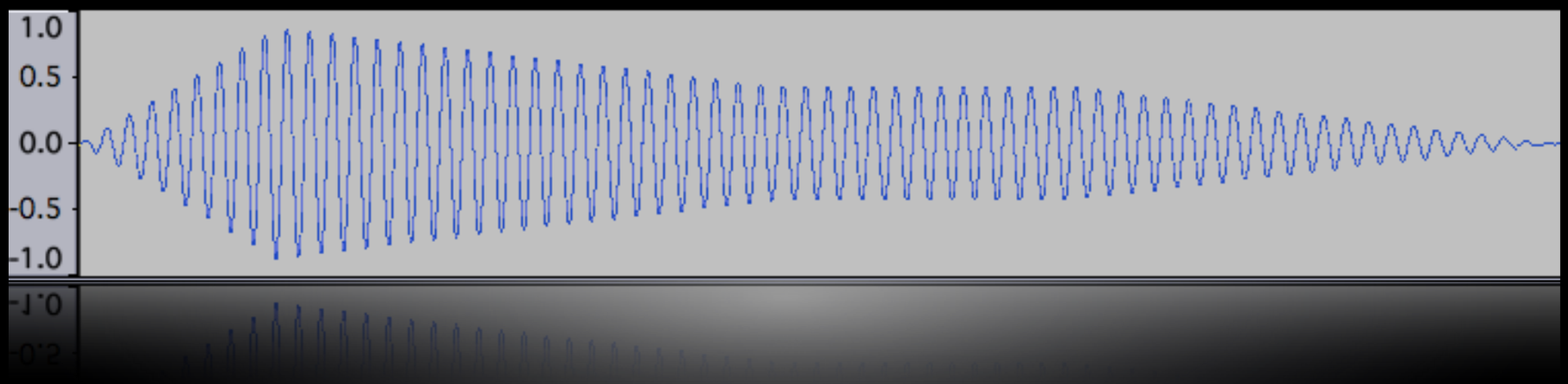
アタック、ディケイ、サステイン、リリースの値からエンベロープの音量を決定し、★基本となる波との積をとることで、★このような波を得ることができます。シンセサイザーによってはこれら4つのパラメータ以外にもより柔軟なエンベロープを作る為にいくつかのパラメータを持っていることもあります。



×



=

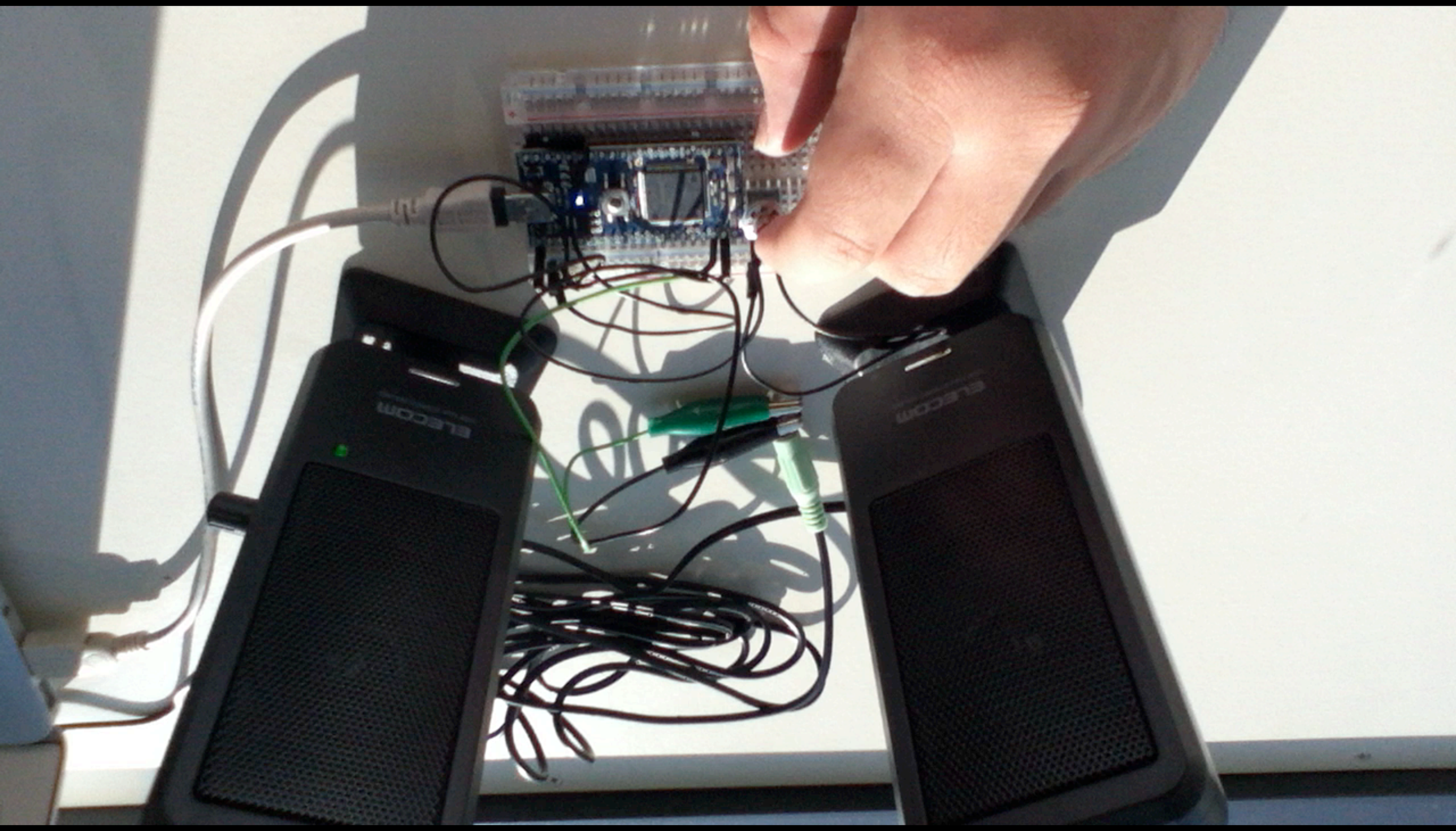


エンベロープ

アタック、ディケイ、サステイン、リリースの値からエンベロープの音量を決定し、★基本となる波との積をとることで、★このような波を得ることができます。シンセサイザーによってはこれら4つのパラメータ以外にもより柔軟なエンベロープを作る為にいくつかのパラメータを持っていることもあります。

```
#include "mbed.h"
AnalogOut audio_out(p18);
DigitalIn button(p19);
class Envelope {
  bool note_stat;
  float prev;
public:
  Envelope() : note_stat( true ), prev( 0.0f ) {}
  void off() { note_stat = false; }
  float operator()( float _time ) {
    if( note_stat ) {
      if( _time < 0.2f ) prev = _time / 0.2f;
      else if( _time < 0.7f ) prev = 1.0f - ( _time - 0.2f );
      else prev = 0.5f;
      return prev;
    }
    else
      return ( prev - _time < 0.0f ) ? 0.0f : prev - _time;
  }
};
int main() {
  while(1) {
    Envelope envelope;
    float time, note_off_time;
    for( time = 0.0f; button; time += 1.0f/16000.0f ) {
      Timer used_time;
      used_time.start();
      audio_out = envelope( time ) * sinf( time * 880.0f * 3.141592f * 2.0f ) * 0.4f + 0.5f;
      used_time.stop();
      wait(1.0f/16000.0f-used_time.read());
    }
    envelope.off();
    for( note_off_time = time; !button; time += 1.0f/16000.0f ) {
      Timer used_time;
      used_time.start();
      audio_out = envelope( time - note_off_time ) * sinf( time * 880.0f * 3.141592f * 2.0f ) * 0.4f + 0.5f;
      used_time.stop();
      wait(1.0f/16000.0f-used_time.read());
    }
  }
}
```

では早速実装してみましょう。★オレンジで示した部分でエンベロープを生成しています。 実際に動かすとこのようになります。

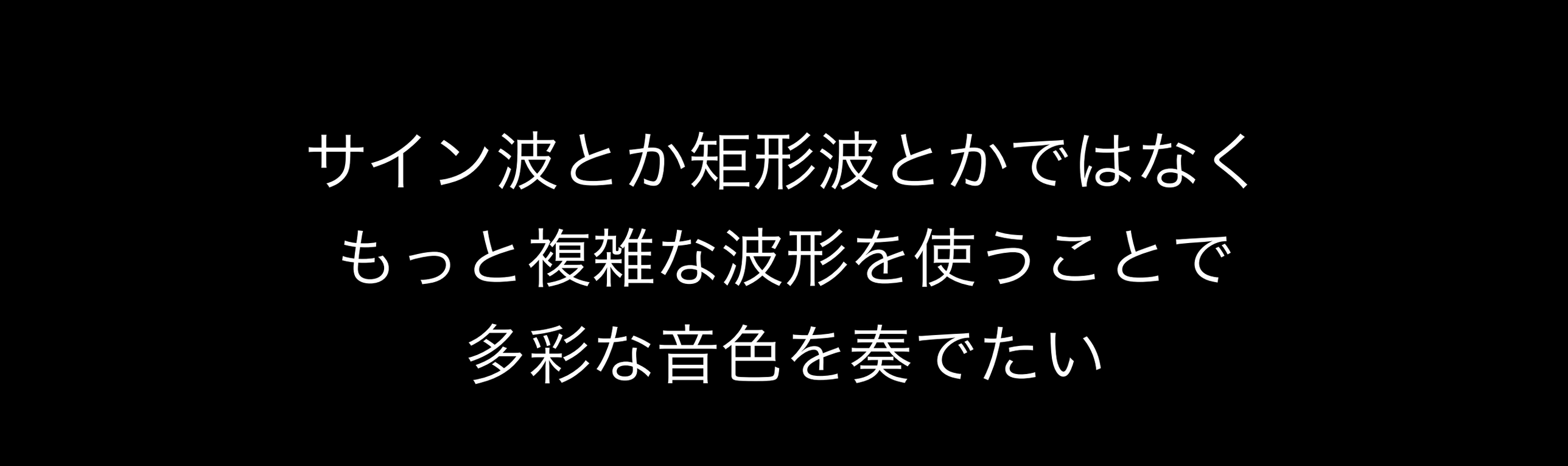


サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

エンベロープを使えば様々な音色を表現できることはわかりましたが、やはりサイン波や矩形波ではなく、もっと複雑な波を使った方が出せる音色の幅が増えるのは確かです。複雑な波形を生成する方法を考えましょう。★作戦1!★ハモンドオルガン★これは複数のサイン波を重ね合わせて複雑な波形を生成しようというものです。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦1



エンベロープを使えば様々な音色を表現できることはわかりましたが、やはりサイン波や矩形波ではなく、もっと複雑な波を使った方が出せる音色の幅が増えるのは確かです。複雑な波形を生成する方法を考えましょう。★作戦1!★ハモンドオルガン★これは複数のサイン波を重ね合わせて複雑な波形を生成しようというものです。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦1

ハモンドオルガン

エンベロープを使えば様々な音色を表現できることはわかりましたが、やはりサイン波や矩形波ではなく、もっと複雑な波を使った方が出せる音色の幅が増えるのは確かです。複雑な波形を生成する方法を考えましょう。★作戦1!★ハモンドオルガン★これは複数のサイン波を重ね合わせて複雑な波形を生成しようというものです。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦1

複数のサイン波を重ね合わせる

ハモンドオルガン

エンベロープを使えば様々な音色を表現できることはわかりましたが、やはりサイン波や矩形波ではなく、もっと複雑な波を使った方が出せる音色の幅が増えるのは確かです。複雑な波形を生成する方法を考えましょう。★作戦1!★ハモンドオルガン★これは複数のサイン波を重ね合わせて複雑な波形を生成しようというものです。

ハモンドオルガン

ハモンドオルガンは1930年代に発明された電気式のオルガンです。ベースとなるサイン波に対して0.5倍、1.5倍、2倍、3倍、4倍、5倍、6倍、8倍の周波数をもつサイン波を、★任意の音量で重ね合わせることでオルガンの音色を作り出します。

これらの倍音を任意の音量で重ね合わせる

ハモンドオルガン

ハモンドオルガンは1930年代に発明された電気式のオルガンです。ベースとなるサイン波に対して0.5倍、1.5倍、2倍、3倍、4倍、5倍、6倍、8倍の周波数をもつサイン波を、★任意の音量で重ね合わせることでオルガンの音色を作り出します。

本物のハモンドオルガンは
サイン波の刻まれた91枚の円盤を
高速回転させることで各周波数の波を作っていた

ハモンドオルガン

ハモンドオルガンが発明された当時、マイクロコントローラなどという便利なものは無かったため、オリジナルのハモンドオルガンでは異なる周波数のサイン波の刻まれた91枚の円盤を高速回転させ、円盤までの距離をピックアップで読み取る事でサイン波を生成していました。★そこまで再現するガッツはないので、マイコンで簡単にサイン波が作れる時代で良かったと思います。

本物のハモンドオルガンは
サイン波の刻まれた91枚の円盤を
高速回転させることで各周波数の波を作っていた

マイコンで簡単にサイン波を作れる時代で

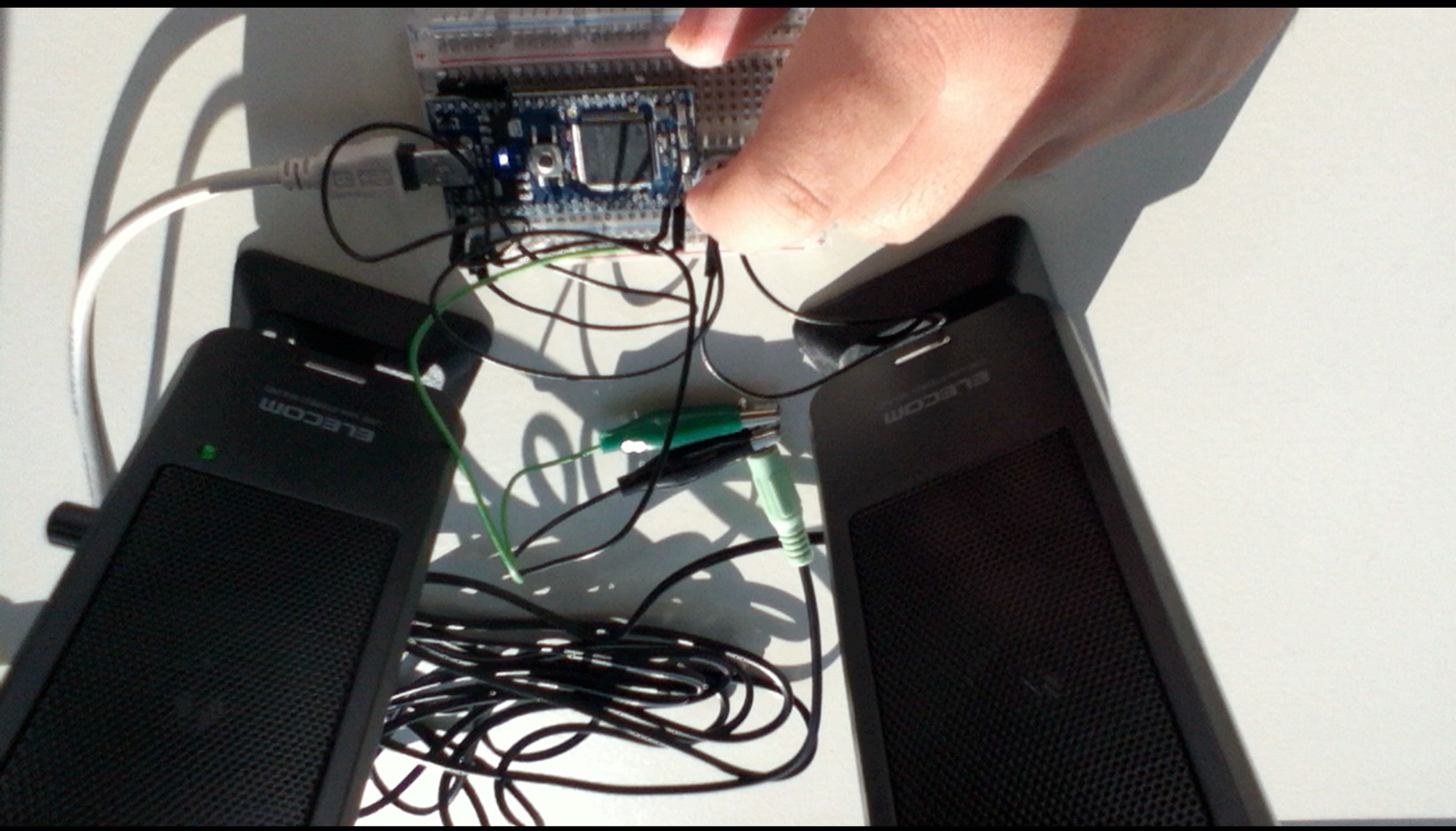
本当に良かった

ハモンドオルガン

ハモンドオルガンが発明された当時、マイクロコントローラなどという便利なものは無かったため、オリジナルのハモンドオルガンでは異なる周波数のサイン波の刻まれた91枚の円盤を高速回転させ、円盤までの距離をピックアップで読み取る事でサイン波を生成していました。★そこまで再現するガッツはないので、マイコンで簡単にサイン波が作れる時代で良かったと思います。

```
...
template< typename Traits >
class Hammond {
public:
  Hammond(){}
  fixed32< 16 > operator()( fixed32< 16 > _time ) {
    static const fixed32< 16 > scale_16 = 220.0f;
    static const fixed32< 16 > scale_8 = 440.0f;
    static const fixed32< 16 > scale_513 = 659.3f;
    static const fixed32< 16 > scale_4 = 880.0f;
    static const fixed32< 16 > scale_223 = 1318.5f;
    static const fixed32< 16 > scale_2 = 1760.0f;
    static const fixed32< 16 > scale_135 = 2217.0f;
    static const fixed32< 16 > scale_113 = 2637.0f;
    static const fixed32< 16 > scale_1 = 3520.0f;
    fixed32< 16 > sum = 0.0f;
    sum += sint( _time * scale_16 ) * Traits::level_16;
    sum += sint( _time * scale_8 ) * Traits::level_8;
    sum += sint( _time * scale_513 ) * Traits::level_513;
    sum += sint( _time * scale_4 ) * Traits::level_4;
    sum += sint( _time * scale_223 ) * Traits::level_223;
    sum += sint( _time * scale_2 ) * Traits::level_2;
    sum += sint( _time * scale_135 ) * Traits::level_135;
    sum += sint( _time * scale_113 ) * Traits::level_113;
    sum += sint( _time * scale_1) * Traits::level_1;
    fixed32< 16 > max = 0.0f;
    max += Traits::level_16;
    max += Traits::level_8;
    max += Traits::level_513;
    max += Traits::level_4;
    max += Traits::level_223;
    max += Traits::level_2;
    max += Traits::level_135;
    max += Traits::level_113;
    max += Traits::level_1;
    sum /= max;
    return sum;
  }
private:
};
...
```

これがハモンドオルガンのコードの抜粋です。★ピンクで示した部分、長いように見えますが単に各周波数のサイン波を足し合わせているだけです。 実際に動かすとこうなります。


ELECOM
ELECOM

問題点

フーリエ級数より
あらゆる波形はサイン波の重ね合わせで作れるが

ハモンドオルガン

ハモンドオルガンの問題点を見てみましょう。フーリエ級数より、あらゆる波形はサイン波の重ね合わせで表現できるのですが、★一般に意図した音色を十分な品質で奏でるために必要なサイン波の数は非常に多く、★たかだか9個のサイン波では大した音色の自由度は得られません。今回は詳しく扱いませんが、このサイン波を大量に重ね合わせるというアプローチを本気でやっていくと、MP3のような音声圧縮技術に辿り着きます。

問題点

フーリエ級数より
あらゆる波形はサイン波の重ね合わせで作れるが

一般に十分な品質を得る為に要求される
サイン波の数は非常に多く

ハモンドオルガン

ハモンドオルガンの問題点を見てみましょう。フーリエ級数より、あらゆる波形はサイン波の重ね合わせで表現できるのですが、★一般に意図した音色を十分な品質で奏でるために必要なサイン波の数は非常に多く、★たかだか9個のサイン波では大した音色の自由度は得られません。今回は詳しく扱いませんが、このサイン波を大量に重ね合わせるというアプローチを本気でやっていくと、MP3のような音声圧縮技術に辿り着きます。

問題点

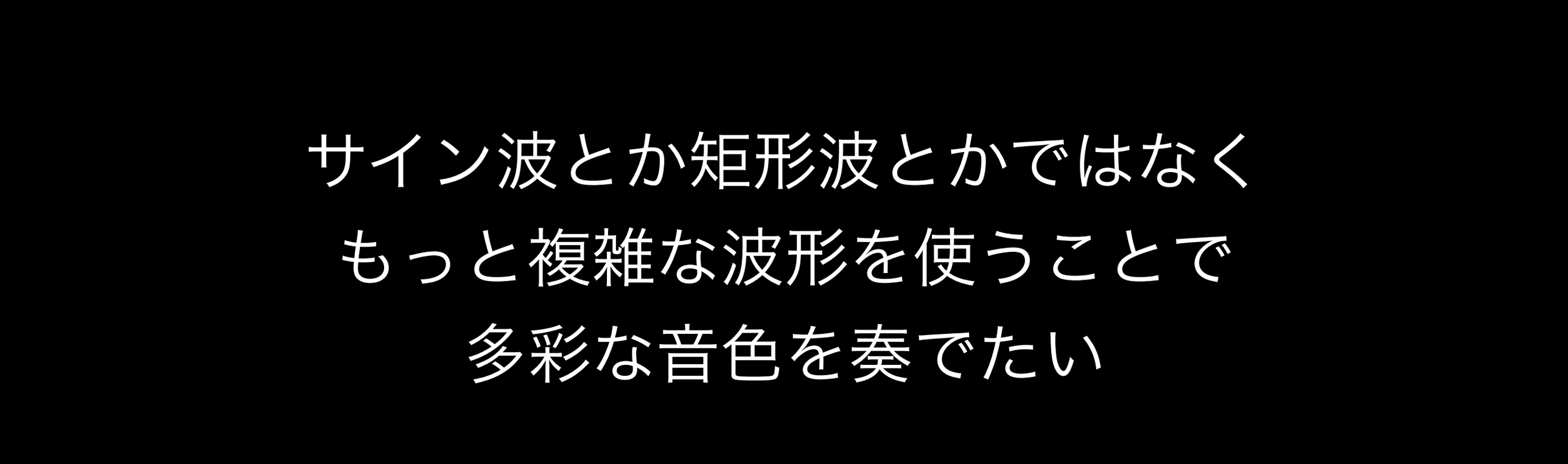
フーリエ級数より
あらゆる波形はサイン波の重ね合わせで作れるが

一般に十分な品質を得る為に要求される
サイン波の数は非常に多く

たかだか9個程度のサイン波では
大した音色の自由度は得られない

ハモンドオルガン

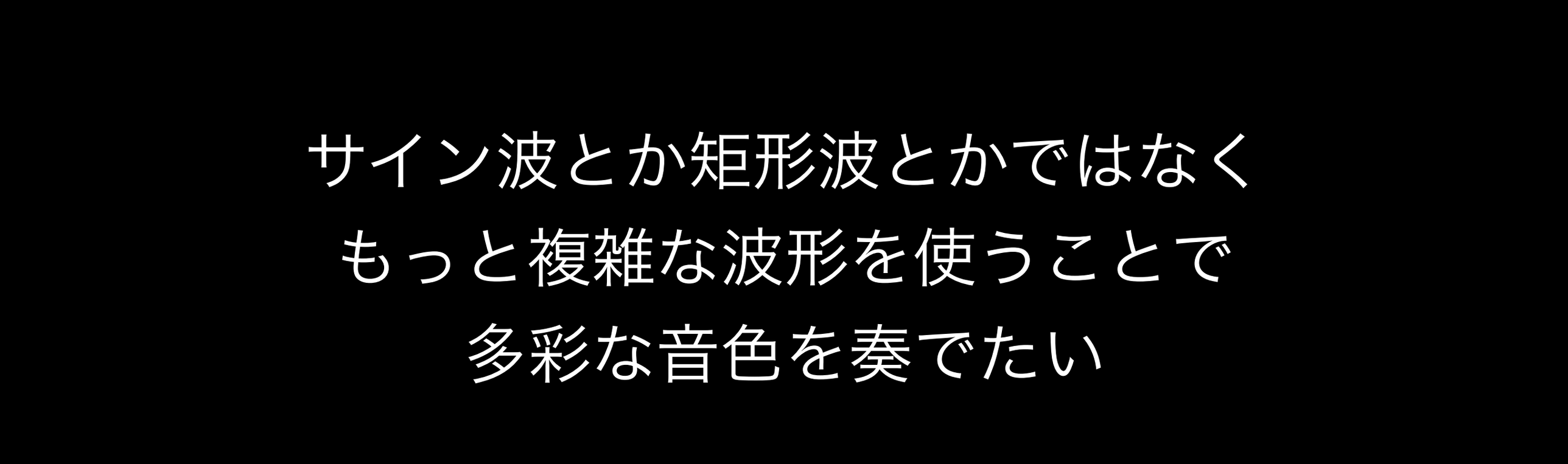
ハモンドオルガンの問題点を見てみましょう。フーリエ級数より、あらゆる波形はサイン波の重ね合わせで表現できるのですが、★一般に意図した音色を十分な品質で奏でるために必要なサイン波の数は非常に多く、★たかだか9個のサイン波では大した音色の自由度は得られません。今回は詳しく扱いませんが、このサイン波を大量に重ね合わせるというアプローチを本気でやっていくと、MP3のような音声圧縮技術に辿り着きます。



別の方法を見てみましょう。★おっさんホイホイへようこそ、FM音源です。★FM音源は名前の通りFM変調を使ってサイン波を歪めることによって複雑な音色を作り出します。かつてPCやゲーム機によく搭載されていたので、80年代のマイコンブームをご存知の方なら、きっと聞いたことがあると思います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦2



別の方法を見てみましょう。★おっさんホイホイへようこそ、FM音源です。★FM音源は名前の通りFM変調を使ってサイン波を歪めることによって複雑な音色を作り出します。かつてPCやゲーム機によく搭載されていたので、80年代のマイコンブームをご存知の方なら、きっと聞いたことがあると思います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦2

FM音源

別の方法を見てみましょう。★おっさんホイホイへようこそ、FM音源です。★FM音源は名前の通りFM変調を使ってサイン波を歪めることによって複雑な音色を作り出します。かつてPCやゲーム機によく搭載されていたので、80年代のマイコンブームをご存知の方なら、きっと聞いたことがあると思います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦2

FM変調を使ってサイン波を歪める

FM音源

別の方法を見てみましょう。★おっさんホイホイへようこそ、FM音源です。★FM音源は名前の通りFM変調を使ってサイン波を歪めることによって複雑な音色を作り出します。かつてPCやゲーム機によく搭載されていたので、80年代のマイコンブームをご存知の方なら、きっと聞いたことがあると思います。

オペレータ

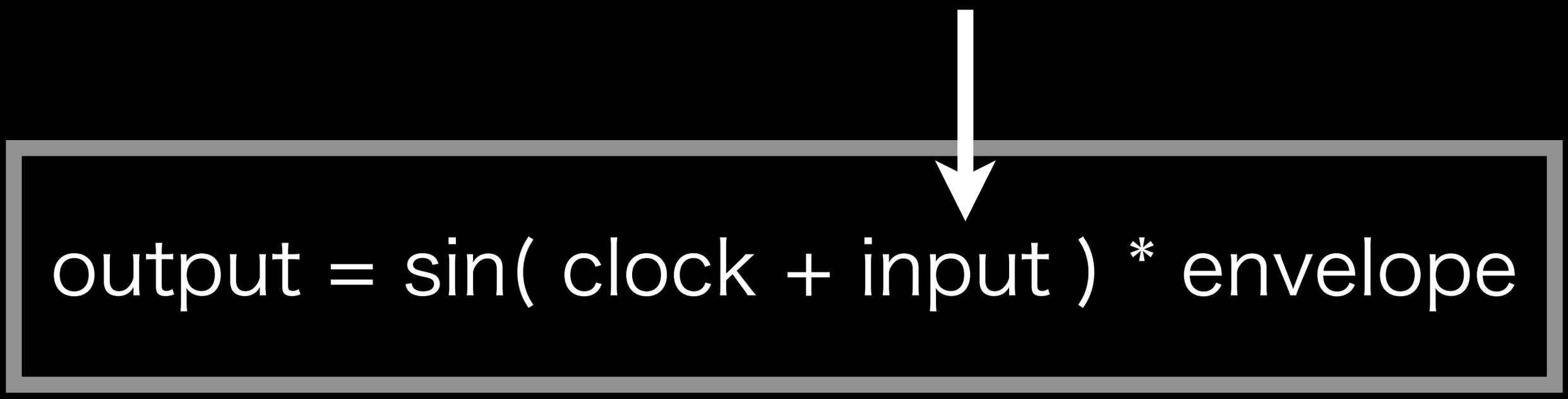
FM音源ではオペレータと呼ばれるブロックを繋ぎ合わせることで音を作っていきます。★オペレータはこのような計算を行います。★オペレータはまず0または別のオペレータからの値を受け取り、それとクロックを足し合わせてsin関数にかけます。つまり、入力側に繋がっているオペレータの値の分だけ、ずれた時刻のサイン波の値が得られます。★そして、それにエンベロープを適用したものをオーディオ出力か、あるいは別のオペレータに対して出力します。

output = sin(clock + input) * envelope

オペレータ

FM音源ではオペレータと呼ばれるブロックを繋ぎ合わせることで音を作っていきます。★オペレータはこのような計算を行います。★オペレータはまず0または別のオペレータからの値を受け取り、それとクロックを足し合わせてsin関数にかけます。つまり、入力側に繋がっているオペレータの値の分だけ、ずれた時刻のサイン波の値が得られます。★そして、それにエンベロープを適用したものをオーディオ出力か、あるいは別のオペレータに対して出力します。

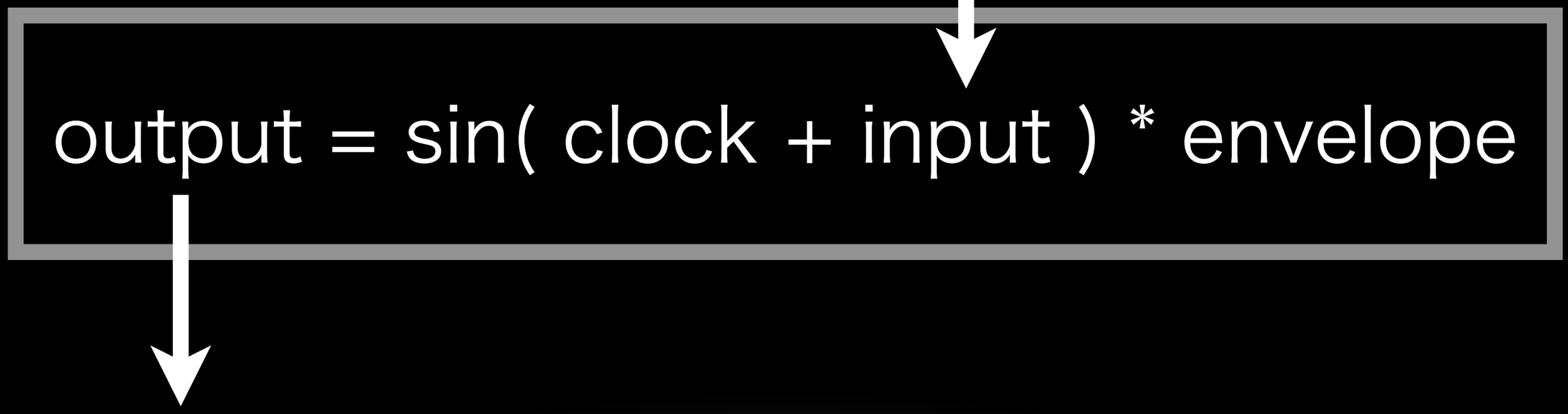
0または別のオペレータ



オペレータ

FM音源ではオペレータと呼ばれるブロックを繋ぎ合わせることで音を作っていきます。★オペレータはこのような計算を行います。★オペレータはまず0または別のオペレータからの値を受け取り、それとクロックを足し合わせてsin関数にかけます。つまり、入力側に繋がっているオペレータの値の分だけ、ずれた時刻のサイン波の値が得られます。★そして、それにエンベロープを適用したものをオーディオ出力か、あるいは別のオペレータに対して出力します。

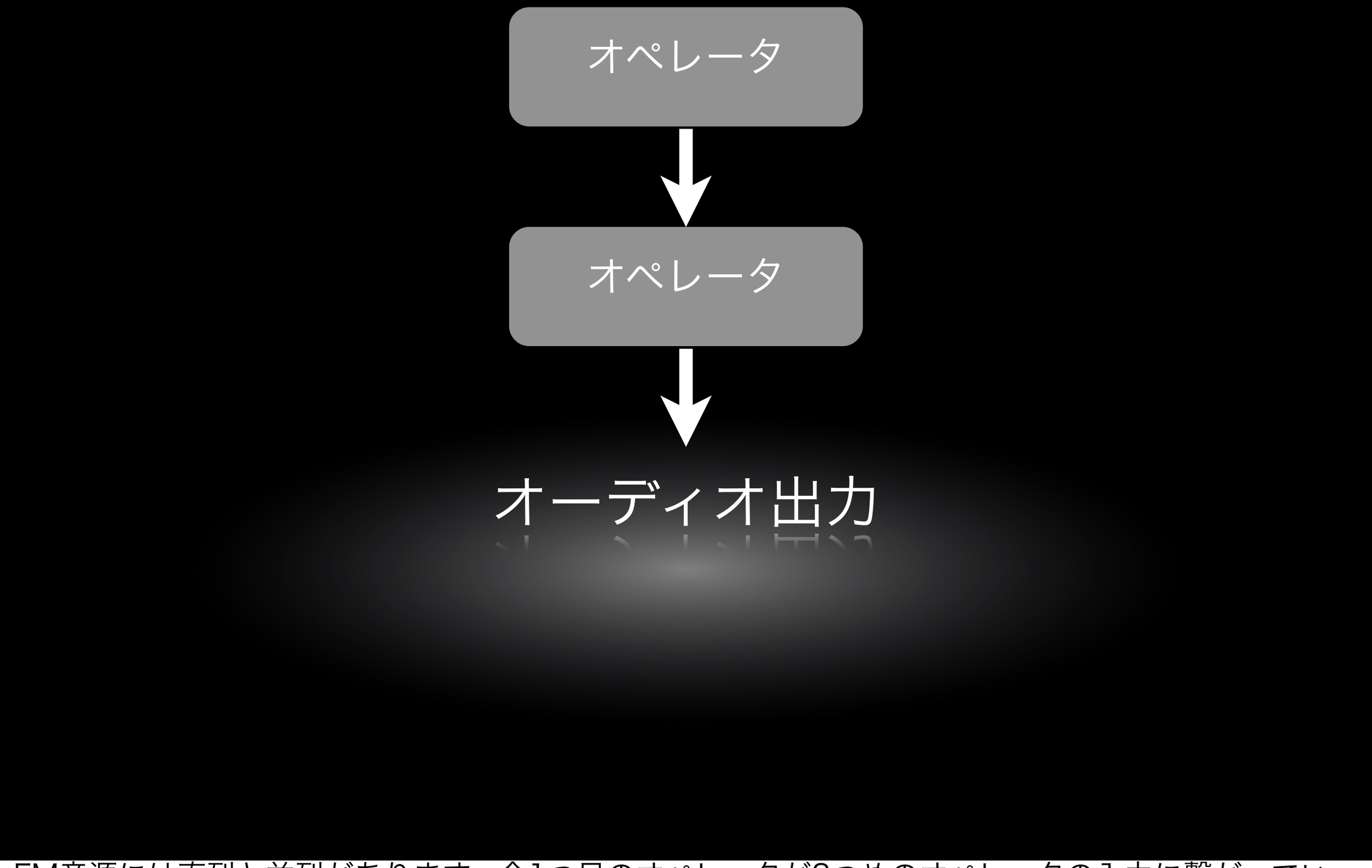
0または別のオペレータ



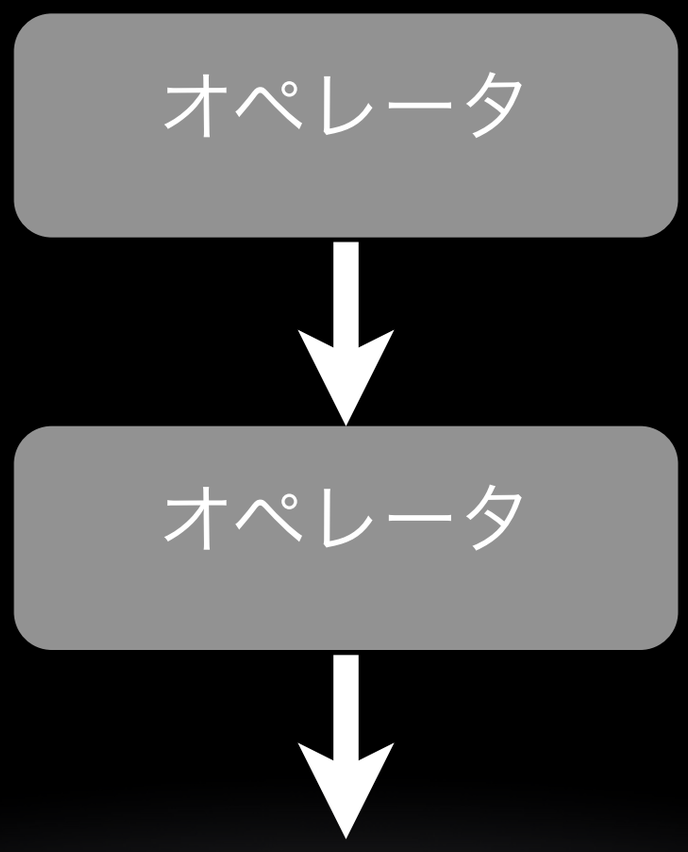
オーディオ出力または別のオペレータ

オペレータ

FM音源ではオペレータと呼ばれるブロックを繋ぎ合わせることで音を作っていきます。★オペレータはこのような計算を行います。★オペレータはまず0または別のオペレータからの値を受け取り、それとクロックを足し合わせてsin関数にかけます。つまり、入力側に繋がっているオペレータの値の分だけ、ずれた時刻のサイン波の値が得られます。★そして、それにエンベロープを適用したものをオーディオ出力か、あるいは別のオペレータに対して出力します。



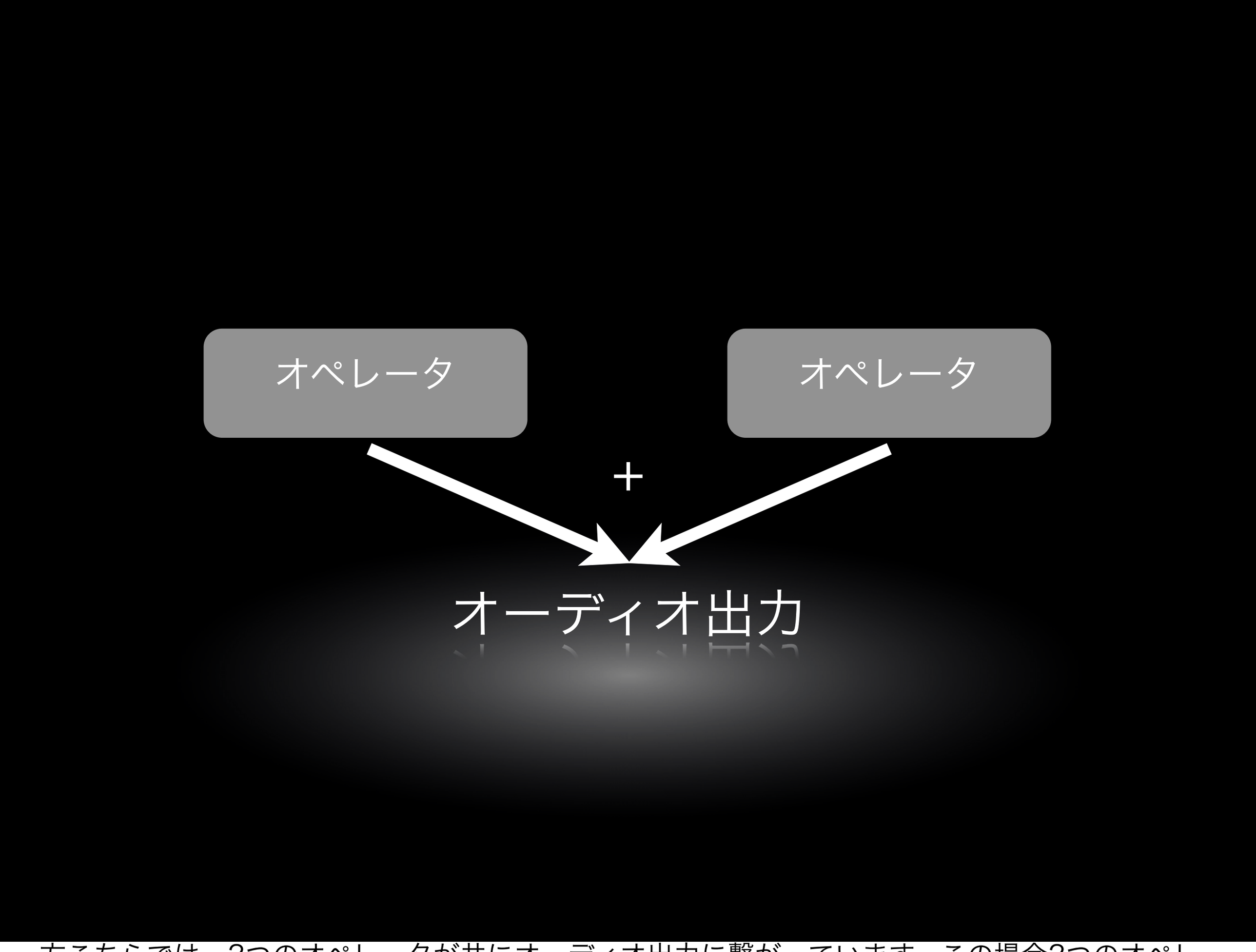
FM音源には直列と並列があります。今1つ目のオペレータが2つめのオペレータの入力に繋がっていて、2つめのオペレータの出力がオーディオ出力に繋がっています。この状態をFM音源では、★2オペレータ直列のFM音源と呼びます。



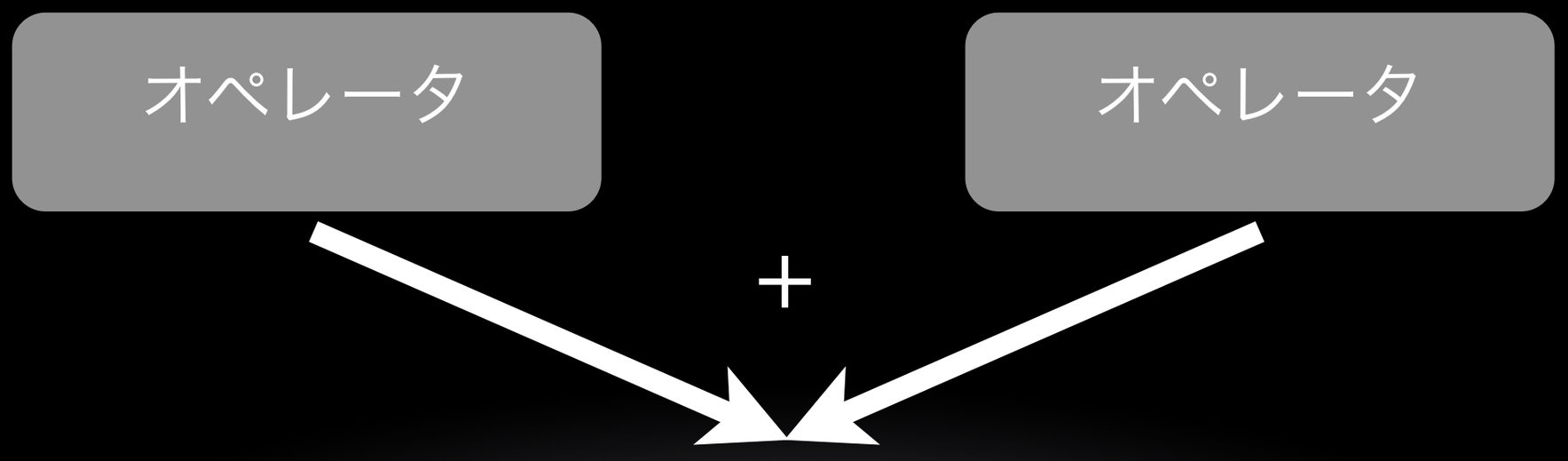
オーディオ出力

2オペレータ直列のFM音源

FM音源には直列と並列があります。今1つ目のオペレータが2つめのオペレータの入力に繋がっていて、2つめのオペレータの出力がオーディオ出力に繋がっています。この状態をFM音源では、★2オペレータ直列のFM音源と呼びます。



一方こちらでは、2つのオペレータが共にオーディオ出力に繋がっています。この場合2つのオペレータの出力は単純に加算されます。この状態のことを、★2オペレータ並列のFM音源と呼びます。入力に何も繋がれていないオペレータからはサイン波が出るため、ハモンドオルガンは9オペレータ並列のFM音源とみなすことも出来ます。



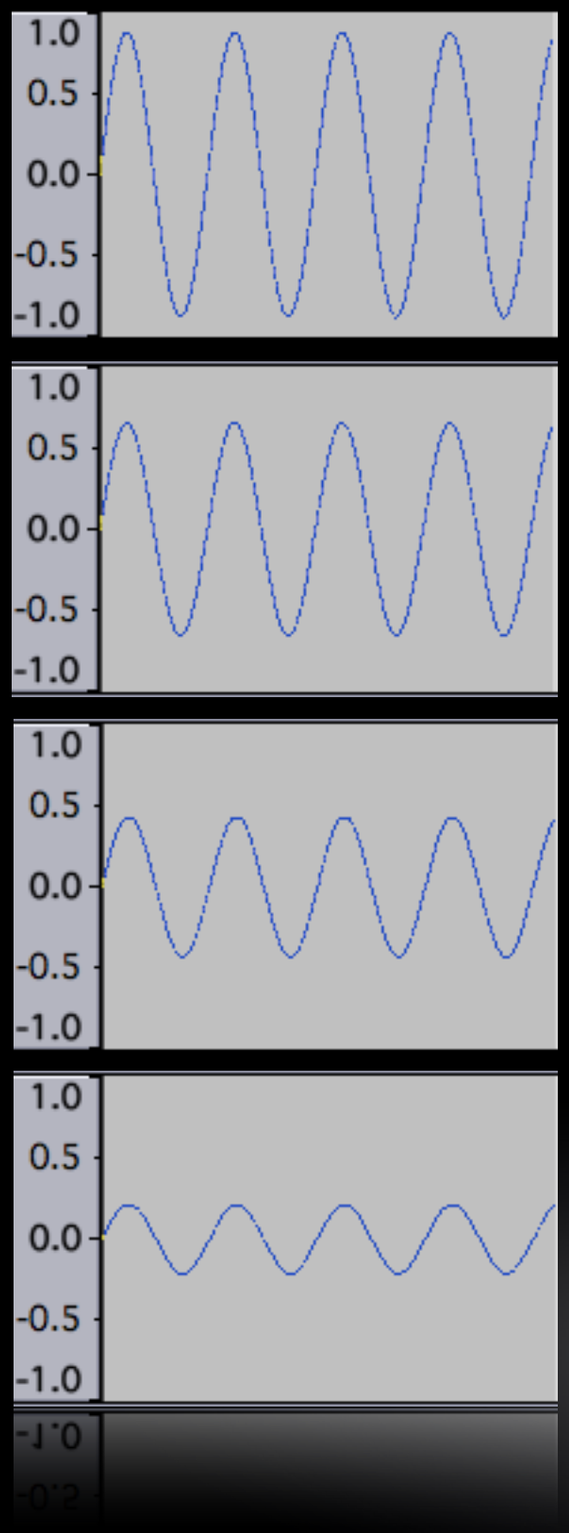
オーディオ出力

2オペレータ並列のFM音源

一方こちらでは、2つのオペレータが共にオーディオ出力に繋がっています。この場合2つのオペレータの出力は単純に加算されます。この状態のことを、★2オペレータ並列のFM音源と呼びます。入力に何も繋がれていないオペレータからはサイン波が出るため、ハモンドオルガンは9オペレータ並列のFM音源とみなすことも出来ます。

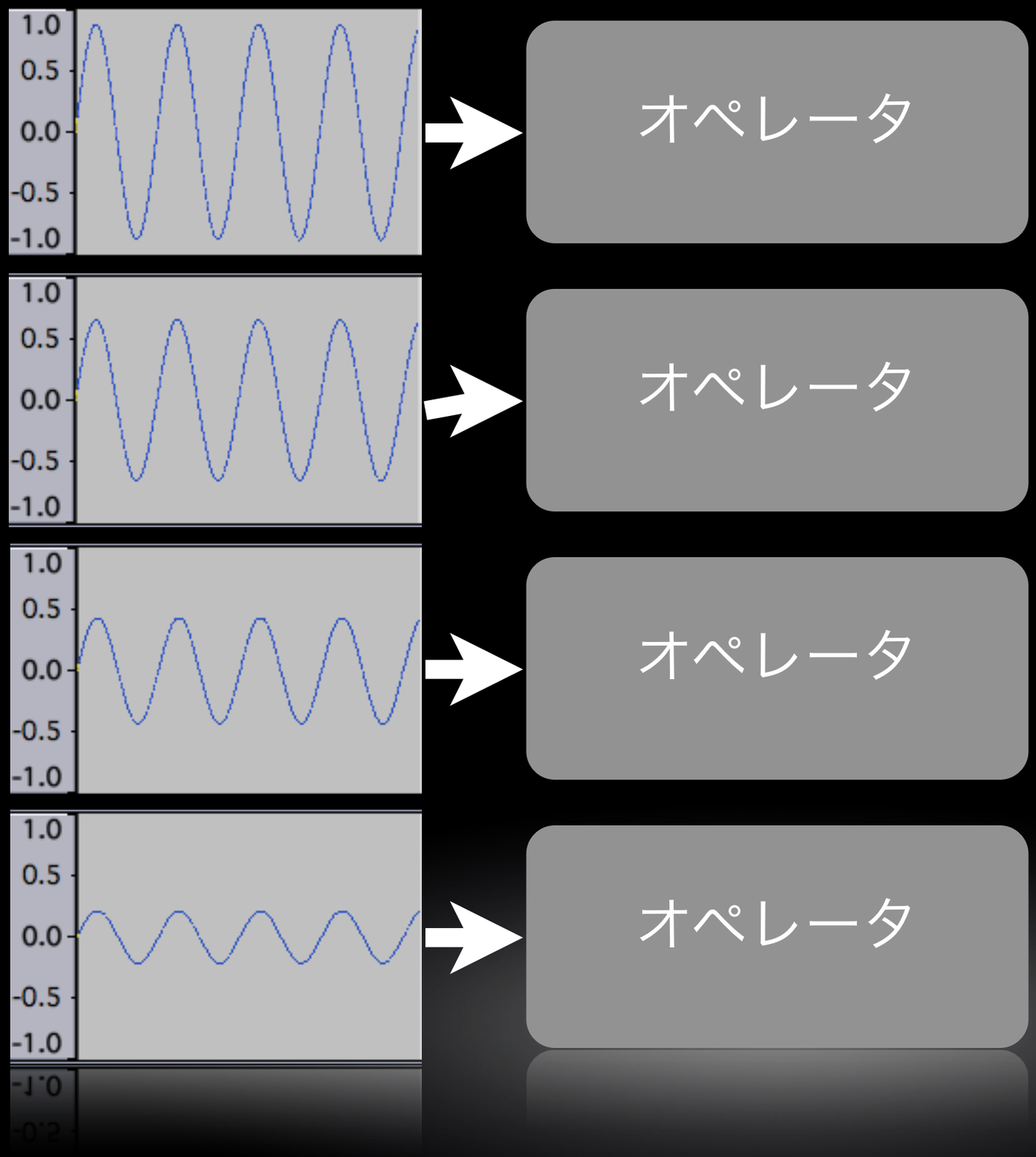
入力側の音量で歪み具合が調節できる

★今このようなサイン波が★オペレータに入力されました。★その結果はこのようになります。左側の4つのサイン波の違いは音量だけですが、右側ではその違いはサイン波の歪みの強さに反映されています。このように、FM音源では入力側の音量によって出力側の歪み具合を調節することができます。

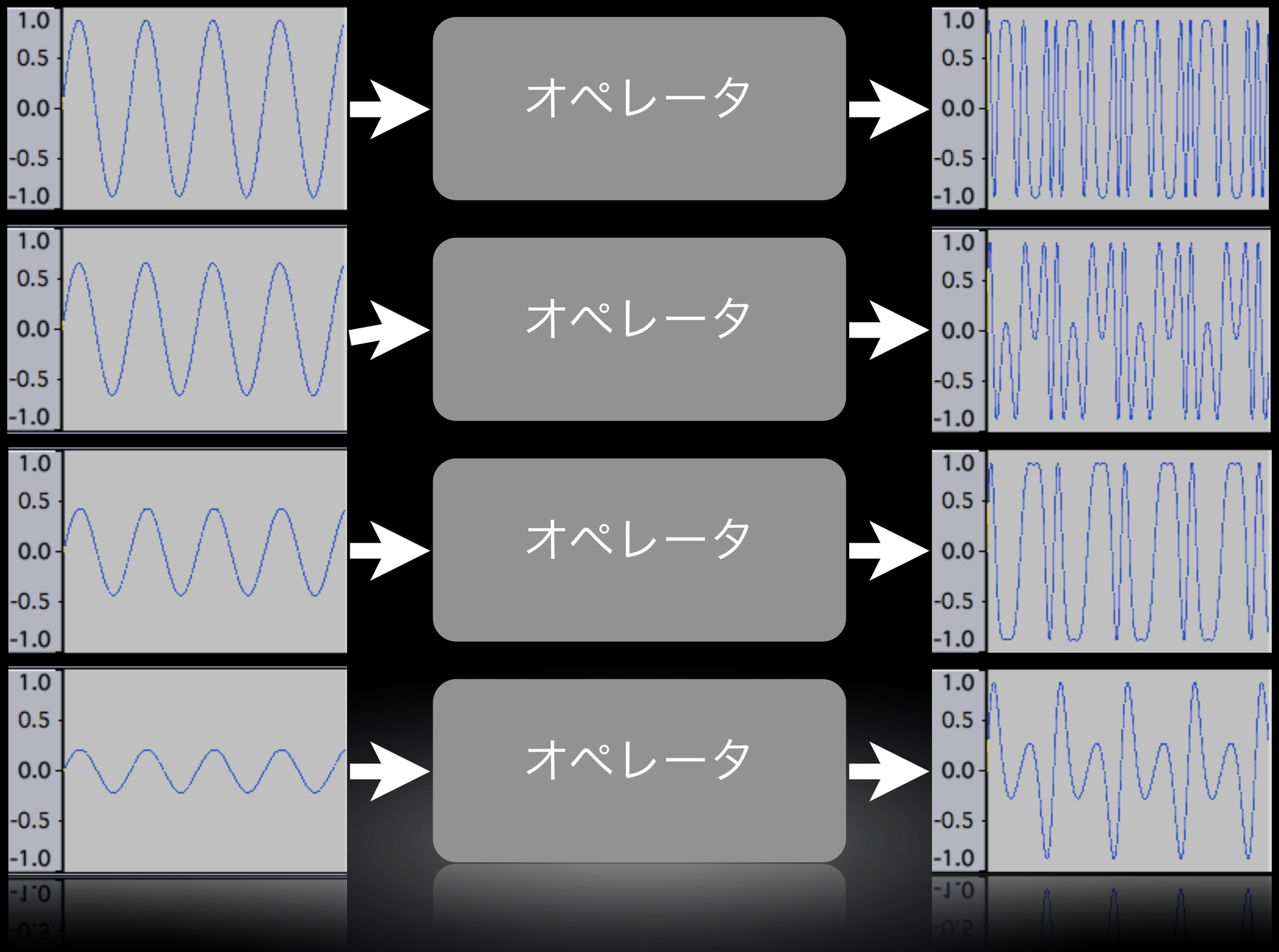


入力側の音量で歪み具合が調節できる

★今このようなサイン波が★オペレータに入力されました。★その結果はこのようになります。左側の4つのサイン波の違いは音量だけですが、右側ではその違いはサイン波の歪みの強さに反映されています。このように、FM音源では入力側の音量によって出力側の歪み具合を調節することができます。



★今このようなサイン波が★オペレータに入力されました。★その結果はこのようになります。左側の4つのサイン波の違いは音量だけですが、右側ではその違いはサイン波の歪みの強さに反映されています。このように、FM音源では入力側の音量によって出力側の歪み具合を調節することができます。



★今このようなサイン波が★オペレータに入力されました。★その結果はこのようになります。左側の4つのサイン波の違いは音量だけですが、右側ではその違いはサイン波の歪みの強さに反映されています。このように、FM音源では入力側の音量によって出力側の歪み具合を調節することができます。

output = sin(clock + input) * envelope

入力側の音量で歪み具合が調節できる

ところで、先ほど見たFM音源のオペレータには★エンベロープが付いていましたよね。エンベロープは音量を時間変化させるものです。★つまり直接オーディオ出力に繋がっていないオペレータのエンベロープは、波形の歪みを時間変化させるために使うことができます。

output = sin(clock + input) * envelope

オペレータには
エンベロープがついている

入力側の音量で歪み具合が調節できる

ところで、先ほど見たFM音源のオペレータには★エンベロープが付いていましたよね。エンベロープは音量を時間変化させるものです。★つまり直接オーディオ出力に繋がっていないオペレータのエンベロープは、波形の歪みを時間変化させるために使うことができます。

output = sin(clock + input) * envelope

オペレータには
エンベロープがついている

直接オーディオ出力に繋がっていない

オペレータのエンベロープは
波形の歪みを時間変化させるために使える

入力側の音量で歪み具合が調節できる

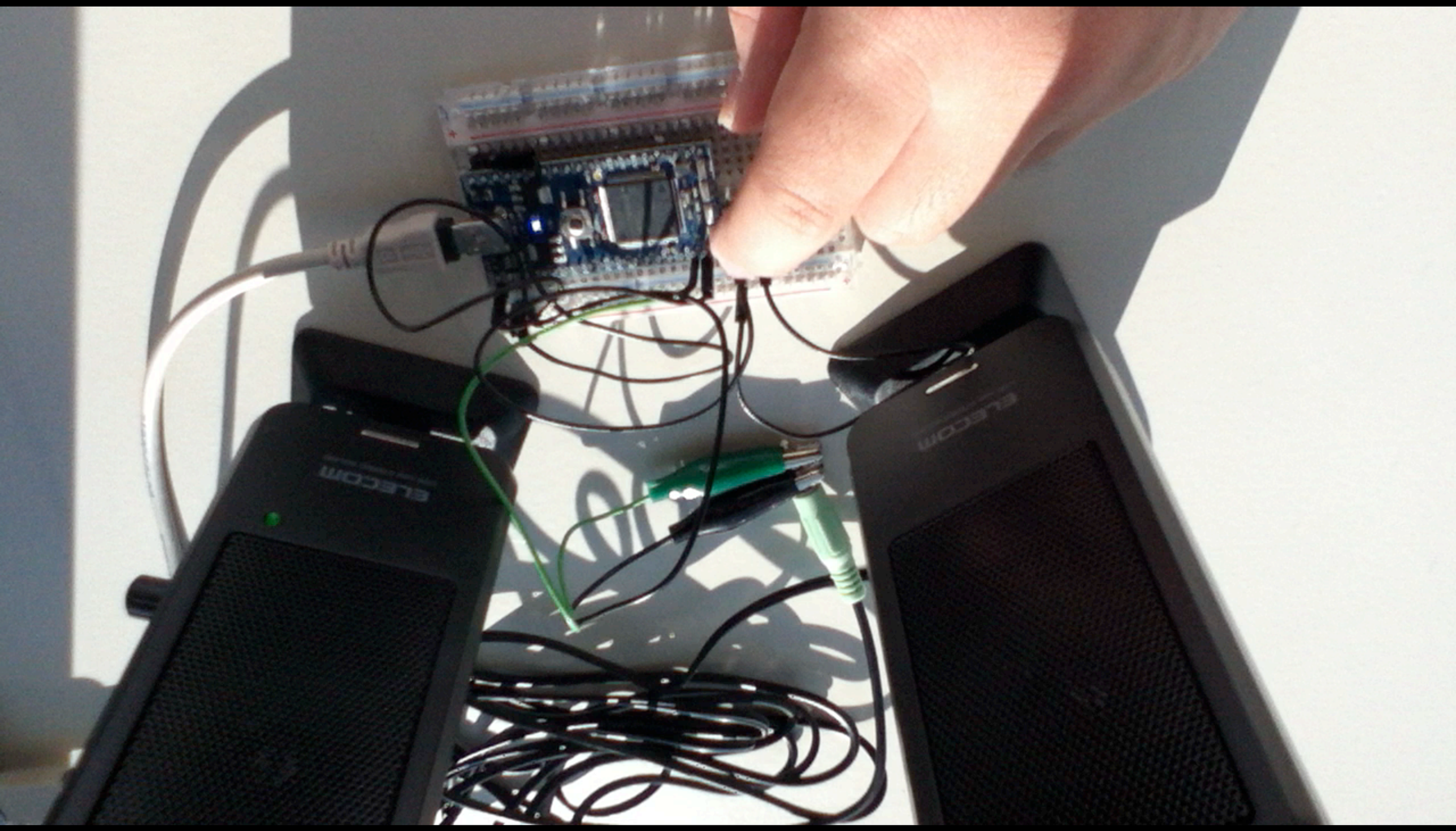
ところで、先ほど見たFM音源のオペレータには★エンベロープが付いていましたよね。エンベロープは音量を時間変化させるものです。★つまり直接オーディオ出力に繋がっていないオペレータのエンベロープは、波形の歪みを時間変化させるために使うことができます。

```
...
class Const {
public:
  fixed32< 16 > operator()( fixed32< 16 > _time ) {
    static const fixed32< 16 > value = 0;
    return value;
  }
};

template< typename Source >
class FM {
public:
  FM() {}
  void off( fixed32< 16 > _off_time ) { envelope.off( _off_time ); }
  fixed32< 16 > operator()( fixed32< 16 > _time ) {
    fixed32< 16 > looped_time = _time - static_cast< int >( _time );
    return sint( ( looped_time * 880.0f + source( _time ) / 2 ) ) * envelope( _time );
  }
private:
  Source source;
  Envelope envelope;
};

int main() {
  while(1) {
    FM< FM< Const > > fm;
    fixed32< 16 > time, note_off_time;
    for( time = 0.0f; button; time += 1.0f/16000.0f ) {
      Timer used_time;
      used_time.start();
      audio_out = fm( time ) * 0.4f + 0.5f;
      used_time.stop();
      wait(1.0f/16000.0f-used_time.read());
    }
    fm.off( time );
...
```

これはFM音源のコードの抜粋です。黄色の部分がオペレータの計算を行っている部分です。 実際に動かすとこんな音が出ます。



問題点

FM音源

FM音源の問題点を見てみましょう。★FM変調の結果は直感的に予想し辛いので、★意図した音色を作る為には多くの場合試行錯誤が必要になります。

問題点

FM変調の結果は直感的に予想し辛いため

FM音源

FM音源の問題点を見てみましょう。★FM変調の結果は直感的に予想し辛いので、★意図した音色を作る為には多くの場合試行錯誤が必要になります。

問題点

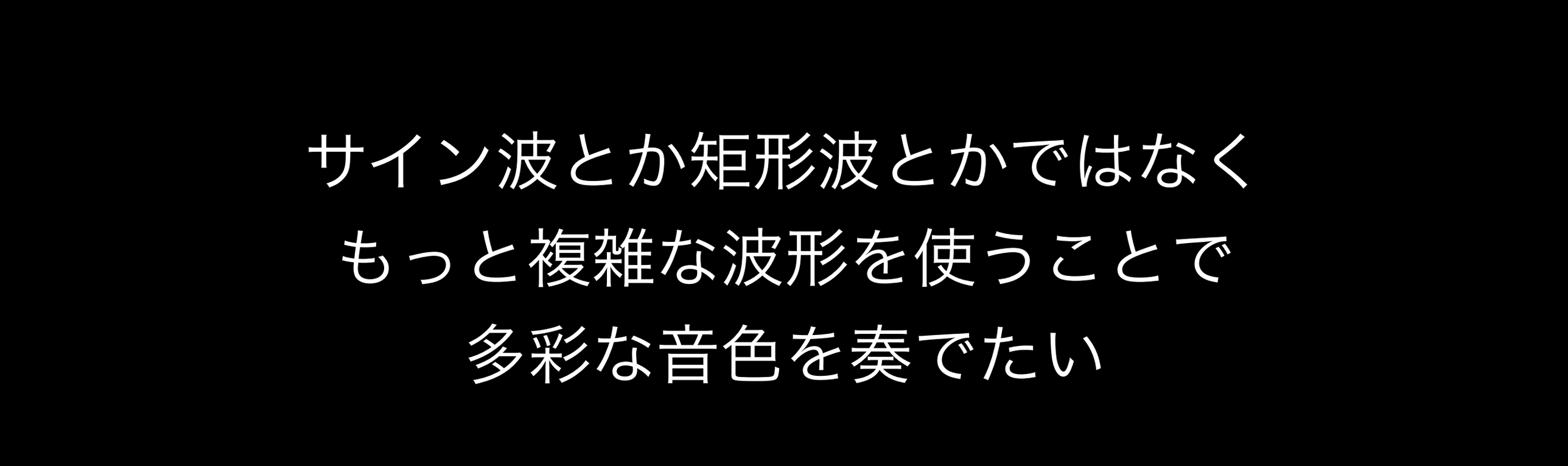
FM変調の結果は直感的に予想し辛いため

意図した音色を作るために試行錯誤が必要

FM音源

FM音源の問題点を見てみましょう。★FM変調の結果は直感的に予想し辛いので、★意図した音色を作る為には多くの場合試行錯誤が必要になります。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい



作戦3!★波形テーブル音源★事前に用意した波形データを使います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦3

作戦3!★波形テーブル音源★事前に用意した波形データを使います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦3

波形テーブル音源

作戦3!★波形テーブル音源★事前に用意した波形データを使います。

サイン波とか矩形波とかではなく
もっと複雑な波形を使うことで
多彩な音色を奏でたい

作戦3

事前に用意した波形データを使う

波形テーブル音源

作戦3!★波形テーブル音源★事前に用意した波形データを使います。

技術的には面白くないけど
それなりに良い音が出る方法

波形テーブル音源

この方法は技術的にはあまり面白くないのですが、それなりに良い音が出る事から広く使われています。★まず、本物の楽器から音を録音してきます。★そしてそれを鳴らしたい音階の周波数に応じたはやさでループ再生します。

技術的には面白くないけど
それなりに良い音が出る方法

配列

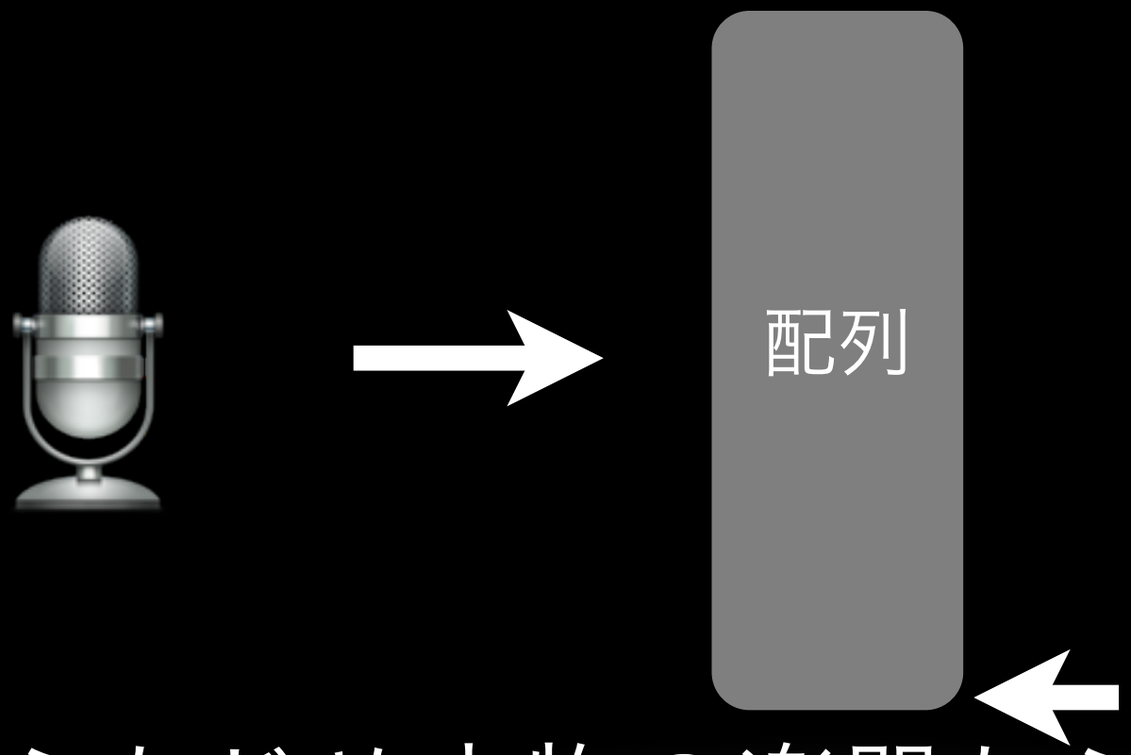
あらかじめ本物の楽器から録音してきた

波形データを音階に対応するはやさで舐める

波形テーブル音源

この方法は技術的にはあまり面白くないのですが、それなりに良い音が出る事から広く使われています。★まず、本物の楽器から音を録音してきます。★そしてそれを鳴らしたい音階の周波数に応じたはやさでループ再生します。

技術的には面白くないけど
それなりに良い音が出る方法



あらかじめ本物の楽器から録音してきた

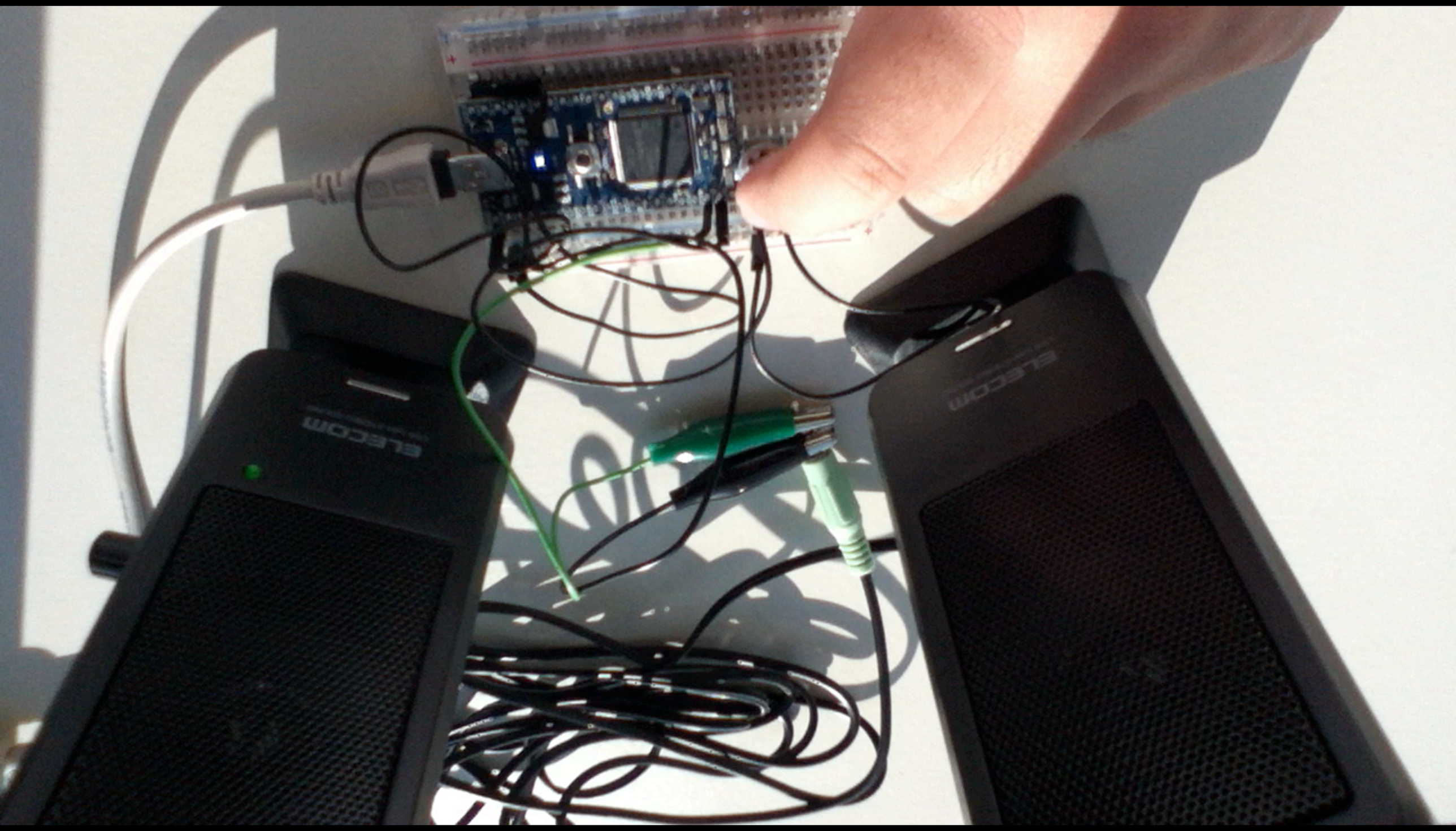
波形データを音階に対応するはやさで舐める

波形テーブル音源

この方法は技術的にはあまり面白くないのですが、それなりに良い音が出る事から広く使われています。★まず、本物の楽器から音を録音してきます。★そしてそれを鳴らしたい音階の周波数に応じたはやさでループ再生します。

```
...
class Guiter {
public:
  virtual fixed32< 16 > operator()( fixed32< 16 > _pos ) {
static const int16_t guiter_table[ 512 ] = {
 -32735,  -32583,  -32431,  -32279,  -32127,  -31975,  -31823,  -31671,
...
 -27519,  -28171,  -28823,  -29475,  -30127,  -30779,  -31431,  -32083,
};
    fixed32< 16 > result;
    fixed32< 16 > a4 = _pos * 220.0f;
    int pos = ( a4.get() >> 7 ) % 512;
//    pos = pos * 880 % 512;
    int32_t value = static_cast< int32_t >( guiter_table[ pos ] ) << 1;
    result.set( value );
    return result * envelope( _pos );
  }
  void off( fixed32< 16 > _off_time ) { envelope.off( _off_time ); }
private:
  Envelope envelope;
};
...
```

波形テーブル音源のコードです。水色で示した部分にギターの音の1周期分の波形が記録されています。 実際に動かすとこうなります。


ELECOM
ELECOM

問題点

波形テーブル音源

問題点を見てみましょう。★波形テーブル音源の欠点はまず時間変化する音色を扱うのが困難であるという点です。これに関しては近年は大容量のメモリが安価になったことから、波形データを長い時間で記録しておく事によってある程度回避されています。★もうひとつは、まず本物の楽器から音を録音してくるという性質上、本物の楽器が存在しない音を作り出すのは困難であるという点です。

問題点

時間変化する音色を扱うのが困難

波形テーブル音源

問題点を見てみましょう。★波形テーブル音源の欠点はまず時間変化する音色を扱うのが困難であるという点です。これに関しては近年は大容量のメモリが安価になったことから、波形データを長い時間で記録しておく事によってある程度回避されています。★もうひとつは、まず本物の楽器から音を録音してくるという性質上、本物の楽器が存在しない音を作り出すのは困難であるという点です。

問題点

時間変化する音色を扱うのが困難

実物の存在しない音を作るのは困難

波形テーブル音源

問題点を見てみましょう。★波形テーブル音源の欠点はまず時間変化する音色を扱うのが困難であるという点です。これに関しては近年は大容量のメモリが安価になったことから、波形データを長い時間で記録しておく事によってある程度回避されています。★もうひとつは、まず本物の楽器から音を録音してくるという性質上、本物の楽器が存在しない音を作り出すのは困難であるという点です。

その他色々実装したけど
そのへんの解説はまたの機会に

ガッツのある人は
ソースを見てみるといいかも

最終的に出来上がったもの

その他にも色々な機能を実装したのですが、それらについてはまた機会があればお話ししましょう。
ガッツのある方はソースコードがgithubの方にアップしてあるので、それを見ると今日説明していないより多くの発見があるかもしれません。それでは最後に出来上がったシンセサイザーのデモをご覧ください。


ELECOM
ELECOM

勉強会復習資料

http://bit.ly/eQKvdE

ソースコード

http://bit.ly/qiaqF9

今日の発表資料は後ほど左のURLに掲載する予定です。また、ソースコード全体を見たい場合は右のURLに置かれているSynthSamplesとTinyFMを参照してください。

ご清聴ありがとうございました

ご清聴ありがとうございました。