大概是v12的格式表，翻译自

<https://osu.ppy.sh/help/wiki/osu!_File_Formats/Osu_(file_format)>

补充自：

https://tieba.baidu.com/p/2515386204?red\_tag=1482394605

黑色为官网翻译

橙色为帖子补充

绿色为自己猜测与注释

红色为疑问项

第一行：地图文件版本

[General]

一些游戏参数

AudioFilename

（String）

指定音频文件位置

AudioLeadIn

(Int 毫秒)

(0-3000)

在音频文件播放前增加的时间，适用于音乐马上就开始的音频文件

PreviewTime

(Int 毫秒)

{-1}(0-?)

在选关界面播放歌曲预览的时间点

Countdown

(Bool)**？**

{0,1}

在第一个Hit出现前是否有倒计时

0=关闭，1=普通，2=慢速，3=快速

SampleSet

(String)

{Normal,Soft,Drum}

在关卡中使用哪种打击音效

Normal=普通音效，Soft=柔和音效，Drum=打击乐音效

StackLeniency

(Float)(时间单位不明)

(0.2-1.0)

相互堆叠的Hit放置时的最小时间差

Mode

(Int)

{0,1,2,3}

本地图的游戏模式

(0=osu!, 1=Taiko, 2=Catch the Beat, 3=osu!mania)

LetterboxInBreaks

(Bool)

在休息的时候是否出现Letterbox窗口(聊天窗口)

WidescreenStoryboard

(Bool)

Storybord(故事模式)是否使用Widescreen(宽屏)

EpilepsyWarning

显示故事模式闪烁警告

CountdownOffset

倒计时延迟时间

StoryFireInFront

显示故事模式于火焰效果前

[Editor]

为制谱者制谱而保存的一些设置

Bookmarks

(Int列表 毫秒)

以逗号分隔的编辑器Bookmarks列表

DistanceSpacing

(float)

Distance Snap的倍率 (默认物品**？**间距)

BeatDivisor

(Int)

{1,2,3,4,6,8,12,16}

放置物品的节拍分割数

音符长短：倒数

(拍号)

GridSize

(Int)

{1,2,4,8}

Grid Snap的网格尺寸

网格大小

(将一拍或小节平均拆分成若干个时间点，每一个hit都放在时间点上，以此保证节拍准确性)

TimelineZoom

(Int)(v14版本中出现了 Float)

编辑器时间线的zoom**？**

[Metadata]

一些音乐和谱面的参数

Title

(String ASCII)

只允许使用ASCII字符组成的音乐标题

TitleUnicode

(String Unicode)

允许使用Unicode的音乐标题，如果没有此项则使用Title

Artist

(String ASCII)

只允许使用ASCII字符组成的作曲家名称

ArtistUnicode

(String Unicode)

允许使用Unicode的作曲家名称，如果没有此项则使用Artist

Creator

(String)

制谱者名称

Version

(String)

谱面的难度名称

Source

(String)

音乐的来源

Tags

(String 列表)

歌曲的Tags，可以在在线列表和选关界面搜索

BeatmapID

(Int)

http://osu.ppy.sh/b/编号

BeatmapSetID

(Int)

http://osu.ppy.sh/s/编号

[Difficulty]

控制谱面难度的一些变量

HPDrainRate

(Float)

(0-10)

扣血量

CircleSize

(Float)

(2-7)

Hit的环的大小

物品大小，数组越小大小越大

OverallDifficulty

(Float)

(0-10)

允许击打一个Hit的时间 (判定容错率)

总体难度：数字越大则击打判断区间越小

ApproachRate

(Float)

(0-10)

从环出现到物品出现(击打点？)所用的时间

出现物品速度：数字越大则出现速度越快

SliderMultiplier

(Float)

滑条速度倍率，默认是1.4

SliderTickRate

(Float)[数字]**(整数？)**

滑条中小点的出现频率，默认是1

每拍滑条小点个数

[Events]

故事模式的事件表

//故事模式的代码比较复杂 <https://osu.ppy.sh/help/wiki/Storyboard_Scripting>

//特别需要处理的部分是背景加载和休息点

//如果检测到0,0,”(String)”,x,y的格式是背景，第三个参数是背景的文件名，第四五个参数是坐标

//如果检测到2,a,b的格式是休息点，第二个参数是起始点，第三个参数是终止点 单位是毫秒

//目前正在编写的文件阅读方法(2018.2.8)会将此部分以行拆分后再以逗号拆分并以String的形式读入String数组的ArrayList中，注意使用的时候转int

[TimingPoints]

谱面的时间点和打击音效表，时间点可以用于修改bpm和音效等信息

结构：Offset, Milliseconds per Beat, Meter, Sample Set, Sample Index, Volume, Inherited, Kiai Mode

Offset

(Int 毫秒)

这个TimingPoint的所在时间点，TimingPoing声明的信息的有效期是从当前的Offset到下一个Offset，第一个时间点从0开始，所以第一个时间点的Offset值是被忽略的，在这个部分时间点需要严格按照Offset值排序

Milliseconds per Beat

(Float)

每一拍经过的毫秒数，当数值为正时表示直接修改此值，当数值为负时，表示修改为前一个非负值的百分比，例如三个连续的TimingPoint的此值为500，-50，-100，则实际效果为0.5秒一拍，0.25秒一拍，0.5秒一拍(没有说明取0，可能是能够保证非0的)

此值=60000/BPM

Meter

(Int)

每小节拍数

Sample Set

Sample Index

(Int)

Sample Set:1=Normal 2=Soft 3=Drum

Sample Index:0=默认

这两个参数是用于索引到自定义音效的

格式见后面音效部分

Volume

(Int 百分之)**?**

(1-100)

音量大小

这三项等待阅读完HitObject后补充

Inherited

(Bool)

TimingPoint的可继承性，Milliseconds per Beat为正数时Inherited为1，为负数时Inherited为0

Kiai Mode

(Bool)

是否开启Kiai Mode，Kiai Mode 开启时，会有一些画面特效，太鼓模式下有加分，详见

<https://osu.ppy.sh/help/wiki/Beatmap_Editor/Kiai_Time>

(可以作为高潮段的一种标志来辅助弹幕生成，但是有的谱面可能就没写Kiai Mode段，所以还是要宏观分析一下)

[Colours]

连击颜色参数(RGB)

结构: Combo [n] : [r],[g],[b]

(Int)

RGB:(0-255)

n必须从1开始连续且已排序

Combo的颜色将按顺序循环

滑条颜色可以被用SliderBody, SliderTrackOverride, SliderBorder三个参数来重写，这些参数是独立于Combo颜色参数的

(尚不清楚是如何界定一个Combo的，可能是一个小节一个combo？)

[HitObjects]

Hit列表

结构: x,y,time,type,hitSound,...,extras

x,y

(Int)

(x:0-512 Pixel y:0-384 Pixel)

(0,0)在屏幕左上角

针对640\*512的尺寸 应该在加上(64,48)来考虑

某些物件，比如转盘的位置是无效的

osu!mania中只有x有效

time

(Int 毫秒)

这个Hit的起始时间

type

(八位二进制)(可能有方法直接处理，如果找不到的话读成int然后处理吧)

type指定了一个HitObject的种类和特性

好像是二进制表示

Bit (0) 点

Bit (1) 滑条

Bit (2) 新Combo

Bit (3) 转盘

Bit (4-6) 组成一个新的数字(0-7)，表示跳过的Combo颜色数量

Bit (7) osu!mania的长条

在阅读的时候会读一份原始数据存到IntUnresolvedType中，然后解析一下将数据拆分存储为：

IntType(0=点，1=滑条，2=转盘，3=mania长条)

IntNewCombo(0=flase,1=true)

IntSkipColour()

点，滑条和转盘不能互相同在，但是可以与新Combo和跳过Combo颜色同在

(同在的原表达为OR’d，可能是指取或，如果理解有误可能是这个词的理解有误)

**还有一段并不是特别理解：**

**The new combo flag always advances to the next combo. The skip value is applied on top of that, so that a skip of 1 means a 2-combo advance. The combo skip value is ignored when the new combo bit is not set.**

**应该是指如果在新Combo处改变颜色，新Combo自带颜色跳颜色，然后等于累加，但是没有理解后面一句话的意思**

**给出的例子是22=2+4+16 一个开启新Combo的滑条并且跳过(Skip)了两个颜色，这个Skip也没有准确理解，是跳过还是前进**

**现在找到的例子里没有使用跳过颜色的，所以还没有去印证**

hitSound

(四位二进制)

指定了成功打击时的音效

Bit(0) 普通音效

Bit(1) 口哨

Bit(2) 谜之乐器(finish 击钹)

Bit(3) 拍手

在阅读的时候会读一份原始数据存到IntUnresolvedHitSound中，然后解析一下将数据拆分存储为：

IntHitSoundNormal(0=flase,1=true)

IntHitSoundWhistle (0=flase,1=true)

IntHitSoundFinish(0=flase,1=true)

IntHitSoundClap(0=flase,1=true)

普通音效总是会被播放，所以Bit(0)是无效的，只有osu!mania中的皮肤的LayeredHitSounds中使用

如果多个位为1则会有音效叠加

如果在TimingPoint中指定了自定义音效列表则按照要求播放自定义音效

自定义音效文件格式

{sample set}-hit{sound}{index}.wav

sample set :normal,soft or drum

sound:normal,whistle,finish or clap

index:自定义音效所在的音效目录，如果不声明则为1

音效的优先顺序：谱面自带>皮肤>默认

extra

额外参数

结构：sampleSet:additionSet:customIndex:sampleVolume:filename

(Int:Int:Int:Int:String)

用于改变音效设置

默认为0:0:0:0:

sampleSet:改变SampleSet到(to)新状态**(不确定是改变到是一直改变还是改变一次)**

0=Auto 从TimingPoint中继承

1=Normal

2=Soft

3=Drum

additionSet

**additionSet (Integer) changes the sample set for the other hit sounds (whistle, finish, clap)未理解**

目前additionSet的值从sampleSet中继承，否则(Otherwise)**(而不是？)**从Timingpoint中继承

customIndex

改变当前的音效目录

sampleVolume

单独音量

(0-100)

音效大小

(取0好像是不改变，默认取0)

Filename

文件地址

单独播放的音效文件地址

相对于地图的文件夹

中间部分

点

没有中间部分

结构: x,y,time,type,hitSound,extras

滑条:

滑条同时表示接水果模式的长条和太鼓模式的长条，但不表示mania中的长条

结构: x,y,time,type,hitSound,sliderType|curvePoints,repeat,pixelLength,edgeHitsounds,edgeAdditions,extras

sliderType

滑条曲线类型

(Char)(大概会按照string存)(实际上按照Int存了：0=L，1=P，2=B，3=C)

{L,P,B,C}

下面会详述各种曲线

curvePoints

若干个点

点的结构:x1:y1|x2:y2|…

曲线

L (Linear)

直线

除起点坐标外有一个点：终点

P(Perfect)

正圆

除起点坐标外有两个点：经过点，终点

为了方便可以转换为一个圆心，一个半径和两个角度

代码

<https://github.com/ppy/osu/blob/master/osu.Game/Rulesets/Objects/CircularArcApproximator.cs>

B(Bézier)

贝塞尔曲线滑条

由一个或多个贝塞尔曲线构成，共享同一个终点(sharing common ends)**(首尾相接？)**

曲线间的角度是任意的

第一个控制点是给出的x,y 其他的点为后面的curvePoints

为了区分两段曲线，曲线间的公共点将会被重复记录

例如

B|437:336|422:309|422:309|384:309|359:337|359:337|328:308|300:304|300:304|272:352|237:383|176:356|159:287|224:256

C(Catmull, deprecated)

可能是CatmullRom差值曲线

应该意思是不推荐使用，但是有些图中可能还存在

官网中并没有给出算法和结构

CatmullRom的资料也比较少

贴吧认为是三次差值曲线，且认为P也是三次差值曲线，不过上文的P是翻译自官网的

Repeat(SliderRepeat)

(Int)

折返次数

1为不折返，2为折返一次，3为折返两次

pixelLength(SliderPixelLength)

(float)

osu坐标下的曲线长度

即一屏为512\*384

这个长度是滑条的真实长度，如果真实长度小于表达式中描述的长度，则路径会收缩**(不确定是提前结束还是比例缩小，不过应该是提前结束，毕竟点都描了)**，如果真实长度大于描述长度，则自然延长，即L直线按直线延长，P弧线按弧延长，B贝赛尔曲线补充一个直线部分(没有说明角度，应该是保持方向延长，那么还需要计算结束方向，活着计算曲线的同时就能得到结束方向)（没有说明如果延长超出屏幕如何操作）

pixelLength用于计算滑条的持续时间，公式如下

slider duration = pixelLength / (100.0 \* SliderMultiplier) \* BeatDuration

SliderMultiplier滑条速度倍率在Difficulty中定义

BeatDuration应该指的是TimingPoints中的Milliseconds per Beat（带倍率修改的当前值）

如果上一句话正确，这个结果的单位是毫秒

贴吧给的公式印证了这一点

贴吧公式：滑条长度\*(600/BPM)/滑条速度=时间长度(毫秒)，应该还要看TimingPoints,有可能给速度设倍数

Milliseconds per Beat=60000/BPM

edgeHitsounds

滑条的声音设置

结构:a|b|c|…

数量等于折返数Repeat(为1不折返)+1

分别对应起点，转折点和结束点的音效设置

应用于滑条的音效只支持normal和whistle

edgeAdditions

滑条的自定义声音设置

结构:a1:b1|a2:b2|…

数量为折返数+1，与edgeHitsounds相对应

a:b结构为sampleSet:additionSet

原理和extra中的是一致的

当sampleSet=0时则首先从extra中继承,其次从Timingpoint中继承

additionSet首先从extra中继承,其次从Timingpoint中继承

Spinners

转盘

结构: x,y,time,type,hitSound,endTime,extras

endTim

(Int 毫秒)

转盘结束时间

打击音效将在转盘结束后播放

osu!mania Hold Notes

mania中的长条

结构: x,y,time,type,hitSound,endTime:extras

应该用不到所以具体算法不翻译了

用到的时候再看

把结构记下不要读关卡的时候报错了