

# Celeste Spinner Analysis - the SPIN theory

**Analysis, research and code by Popax21, with help from XMinty7 and others**

*The name “SPIN theory” is derived from the fact that this theory can be visualized as a recursive set of sliding windows “spinning” / looping around in a residual number field*

## Introduction

Celeste’s spinners can “load” and “unload” (= turn collision on or off by setting `Collidable`) only on certain frames where `Scene.OnInterval(float interval, float offset)` returns `true`. `interval` is always 0.05f for loading and 0.25f for unloading, while `offset` is a random float in  $[0;1)$  chosen when the room is loaded. The following code is a decompilation of `Scene.OnInterval`:

```
public bool OnInterval(float interval, float offset) {  
    return Math.Floor((TimeActive - offset - Engine.DeltaTime) /  
        ⇨ interval) < Math.Floor((TimeActive - offset) / interval);  
}
```

`Scene.OnInterval` is intended to only return `true` for only one frame every `interval` seconds. As such the game uses this additional requirement to limit the number of collidable checks happening every frame.

This behavior leads to spinners being grouped into so called “spinner groups”: the number of spinner groups is determined through the following formula:

$$n_g = \left\lfloor \frac{interval}{dt} \right\rfloor = \lfloor interval * 60 \rfloor$$

Spinners will check whether they should load on every frame  $t = i * n_g + g$  for some integer  $i$ , where  $g \in [0; n_g)$  is called the spinner’s *group (index)*. A “group cycle” consists of  $n_g$  frames, where every spinner checks if it should load exactly once at some frame offset corresponding to its group number  $g$ . The following formula can be used to determine a spinner’s group from its offset (the later

analysis will contain a proof of a more general version of this statement):

$$g = \left\lceil \frac{off_B}{dt} \right\rceil \mod n_g$$

$$\text{where } off_B = \begin{cases} (off \mod intv) - intv & \text{if } (off \mod intv) > intv - dt \\ off \mod intv & \text{else} \end{cases}$$

For the scope of this document, frame indices are defined as the number of frames which have ellapsed since the scene (**=Level**) has become active.

As briefly mentioned above, spinners don't strictly adhere to this group pattern, can actually change which group they belong to at certain points in time - the rest of the document will be an analysis of why and when these spinner group changes occur.

## Why spinner offsets drift

**NOTE:** This analysis assumes that all operations are performed with perfect accuracy. While real-world floating point operations are subject to imprecision, for reasons beyond the scope of this document, the 32 bit floating point operations are actually executed with 80 bits of precision, which means that in practice floating point imprecision can be neglected for **OnInterval**.

Let  $dt$  be the game's delta time (**Engine.DeltaTime** in the code),  $T$  the time since the scene / level has been instantiated (**TimeActive** in the code),  $intv$  and  $off$  the **interval** and **offset** parameters respectively, and  $n_g$  be the number of spinner groups same as above (note that even though  $intv$  is fixed for spinner load groups, this analysis will be a generic analysis for every possible  $intv$  and  $off$ ). Note that because  $T$  is incremented by  $dt$ , we can set  $T = t * dt$ , where  $t$  is the current frame index.

"Translating" the C# code into mathematical notation yields

$$\left\lfloor \frac{T - off - dt}{intv} \right\rfloor < \left\lfloor \frac{T - off}{intv} \right\rfloor$$

which is equivalent to

$$\frac{T - off - dt}{intv} < \left\lfloor \frac{T - off}{intv} \right\rfloor$$

By multiplying both sides by  $intv$  and substituting  $\lfloor \frac{a}{b} \rfloor * b = a - (a \mod b)$  we can reformulate this to obtain:

$$T - off - dt < T - off - ((T - off) \mod intv)$$

Subtracting  $T - off - dt$  from both sides yields

$$-dt < -((T - off) \mod intv) \iff (T - off) \mod intv < dt$$

and because  $intv > dt \iff dt \bmod intv = dt$  we get

$$T - off = t * dt - off < dt \bmod intv$$

For purposes which will become apparent later, let us relax our assumptions slightly from our implicit  $dt > 0$  to  $|dt| > 0$ . This will allow  $dt$  to be negative, which essentially means that time is flowing “backwards” through the cycle, and as such we are stepping through the cycle backwards. Additionally, we will decouple the right hand side of the condition from  $|dt|$  by instead introducing a new symbol  $thr$ , which in this case equals  $thr = |dt|$ . The relaxed statement we will continue to further analyse is:

$$t * dt - off < thr \bmod intv$$

Now, because of spinner groups, we expect this expression to evaluate to the same value for every  $t = c * n_g + i$ , where  $c$  is the index of the group cycle and  $i$  is the offset into the cycle. Inserting this value of  $t$  into the expression yields:

$$t * dt - off = (c * n_g + i) * dt - off = c * n_g * dt + i * dt - off < thr \bmod intv$$

Let us now define the *residual drift*  $r$  as  $r = intv - |dt| * n_g$ , where  $|r| \leq \frac{|dt|}{2}$ . Inserting this new symbol into our condition we get:

$$\begin{aligned} & c * n_g * dt + i * dt - off < thr \bmod intv \\ \iff & c * \text{sgn } dt * (intv - r) + i * dt - off < thr \bmod intv \\ \iff & i * dt - off - \text{sgn } dt * r * c < thr \bmod intv \\ \iff & \mathbf{i} * dt - (off + \text{sgn } dt * \mathbf{r} * \mathbf{c}) < thr \bmod intv \end{aligned}$$

Note that the final expression differs from what we would expect if all spinner cycles were exactly equal! Instead of  $i * dt - off < thr \bmod intv$ , which would mean that the values of our expression would loop every group cycle (as it is only dependent on the group offset  $i$ ), we got  $i * dt - (off + \text{sgn } dt * r * c) < thr \bmod intv$ . **This means that the effective offset of each spinner shifts by  $\text{sgn } dt * r$  every  $n_g$  frames!**

## When drift happens - the check range

Let us start by finding all values  $i \in [0; n_g]$  for which  $i * dt - off < thr \bmod intv$  is true. We can differentiate between two cases on the sign of  $dt$ :

- When  $dt > 0$ , then we define

$$off_B = \begin{cases} (off \bmod intv) - intv & \text{if } (off \bmod intv) > intv - dt \\ off \bmod intv & \text{else} \end{cases}$$

(so that  $off = off_B \bmod intv \wedge -dt < off_B \leq intv - dt$ ). Solving for  $i$  yields:

$$\left\lceil \frac{off_B}{dt} \right\rceil = \left\lceil \frac{off_B}{|dt|} \right\rceil \leq i < \left\lceil \frac{off_B}{|dt|} \right\rceil + \left\lceil \frac{thr - (-off_B \bmod |dt|)}{|dt|} \right\rceil$$

- When  $dt < 0$ , then we define

$$off_B = \begin{cases} (off \bmod intv) - intv & \text{if } off \neq 0 \\ 0 & \text{else} \end{cases}$$

(so that  $off = off_B \bmod intv \wedge -intv < off_B \leq 0$ ). We then reformulate the statement as:

$$-off_B - i * |dt| < thr \bmod intv$$

Solving for  $i$  yields:

$$\left\lfloor \frac{-off_B}{|dt|} \right\rfloor - \left\lceil \frac{thr - (-off_B \bmod |dt|)}{|dt|} \right\rceil < i \leq \left\lfloor \frac{-off_B}{|dt|} \right\rfloor = \left\lceil \frac{off_B}{|dt|} \right\rceil$$

(Note that these ranges are not perfectly accurate - there can be group cycles where two or no values of  $i \in [0; n_g)$  will fulfill the condition (namely  $i = 0 \wedge i = n_g - 1$  and  $i = n_g$ ). These are results of drifts which cross cycle boundaries, and these drift-affected solutions of the equation will not be considered when determining spinner's group)

**This means that spinner groups are not actually singlar discrete values, but *ranges of values*** (called the *check range*). We will continue to call  $g = \left\lceil \frac{off_B}{|dt|} \right\rceil \bmod n_g$  the *group* of the spinner, however keep in mind that in practice, the actual check results can differ from what we expect using the simple definition of spinner groups.

**Additionally, this means that spinner check behavior is affected to two types of drift:** drifts of the spinner group  $g$ , called *group drifts*, and temporary anomalies of the length  $L = \left\lceil \frac{thr - (-off_B \bmod |dt|)}{|dt|} \right\rceil$  of the check range, called *length drifts*.

From now on, we will call  $off_B$  the *bounded offset* of the spinner. It will become useful later as  $0 \leq \left\lceil \frac{off_B}{|dt|} \right\rceil * dt - off_B < |dt|$  is true.

## Group drifts

We will disregard  $thr$  by assuming that it always is  $|dt|$  for the rest of the group drift analysis - this eliminates length drift, which could otherwise affect things as well. Check results which disregard  $thr$  like this are called *raw check results*.

Let  $r' = \text{sgn } dt * r$  and  $off_c = off_B + c * r'$ . Assuming our ideal definition of spinner groups, we expect that the following holds for all integer values of  $c$ , where  $0 \leq i * dt - off_{c-1} < |dt|$ :

$$i * dt - off_{c-1} < |dt| \bmod intv \iff i * dt - off_c < |dt| \bmod intv$$

We now want to find all values of  $c$  for which this is not true. Because of  $|r'| \leq \text{intv} - |dt|$ , the right hand side can not be true only because of the modulo term - as such we can restate our earlier condition as:

$$0 \leq v_c < |dt| \iff 0 \leq v_{c-1} = v_c + r' < |dt|$$

(where  $v_c = i * dt - \text{off}_c$ )

This is obviously not always true - we can determine the values of  $c$  for which it won't be by distinguishing between two cases depending on the sign of  $r'$ :

- When  $r' < 0$ , then  $v_c > v_{c-1}$ . As such we want to find all  $c$  for which the following holds:

$$\begin{aligned} & 0 \leq v_{c-1} < |dt| \wedge \neg(0 \leq v_c < |dt|) \\ \iff & 0 \leq v_{c-1} = v_c + r' < |dt| \leq v_c \\ \iff & |dt| \leq v_c < |dt| - r' \\ \implies & v_c = (g * dt - (\text{off}_B + c * r')) < -r' \pmod{|dt|} \\ \iff & -\text{off}_B - c * r' < -r' \pmod{|dt|} \\ \iff & \mathbf{c} * (-r') - \text{off}_B < |r'| \pmod{|dt|} \end{aligned}$$

- As  $|dt| \leq v_c < 2 * |dt|$  holds, the condition is true for  $i_c = i_{c-1} - \text{sgn } dt$  because

$$(i_{c-1} - \text{sgn } dt) * dt - \text{off}_c = i_{c-1} * dt - |dt| - \text{off}_c = v_c - |dt| \wedge 0 \leq v_c - |dt| < |dt|$$

**\* This means that spinner drifts shift the spinner group by  $-\text{sgn } dt = \text{sgn } r$ , and  $i_c$  is not bounded within the range of  $[0; n_g)$ !**

- Note that  $0 \leq i_c * dt - \text{off}_c < |dt|$  still holds.

- When  $r' > 0$ , then  $v_c < v_{c-1}$ . As such we want to find all  $c$  for which the following holds:

$$\begin{aligned} & 0 \leq v_{c-1} < |dt| \wedge \neg(0 \leq v_c < |dt|) \\ \iff & v_c < 0 \leq v_{c-1} = v_c + r' < |dt| \\ \iff & -r' \leq v_c < 0 \\ \iff & 0 \leq v_c + r' < r' \\ \implies & v_c + r' = (g * dt - (\text{off}_B + c * r')) + r' < r' \pmod{|dt|} \\ \iff & -\text{off}_B - c * r' + r' < r' \pmod{|dt|} \\ \iff & \mathbf{c} * (-r') - (\text{off}_B - r') < |r'| \pmod{|dt|} \end{aligned}$$

- As  $-|dt| \leq v_c < 0$  holds, the condition is true for  $i_c = i_{c-1} + \text{sgn } dt$  because

$$(i_{c-1} + \text{sgn } dt) * dt - \text{off}_c = i_{c-1} * dt + |dt| - \text{off}_c = v_c + |dt| \wedge 0 \leq v_c + |dt| < |dt|$$

\* **This means that spinner drifts shift the spinner group by  $\text{sgn } dt = \text{sgn } r$ , and  $i_c$  is not bounded within the range of  $[0; n_g)$ !**

– Note that  $0 \leq i_c * dt - \text{off}_c < |dt|$  still holds.

- When  $r' = 0$ , then there is no group drifting, and as such group changes will never occur

(our initial assumption of  $0 \leq i_c * dt - \text{off}_c < |dt|$  can be shown to inductively hold for all values of  $c$  when  $i_0 = \left\lceil \frac{\text{off}_B}{|dt|} \right\rceil$ , because we know that  $0 \leq \left\lceil \frac{\text{off}_B}{|dt|} \right\rceil * dt - \text{off}_B < |dt|$  holds as our base case, and that when a tick is not affected by drift, then by definition  $0 \leq v_{c-1} < |dt| \iff 0 \leq v_c < |dt|$  holds, or if it is, we have shown that the condition still holds)

**Note how both non-trivial cases match the exact structure of our original statement derived from OnInterval!** This means that the problem is recursive, and the spinner cycle indices where group drift occurs behave just like a recursive instance of the original problem with the following parameters:

$$\begin{aligned}
 dt_R &= -r' = -\text{sgn } dt * r \\
 t_R &= c \\
 \text{intv}_R &= |dt| \\
 \text{off}_R &= \begin{cases} \text{off}_B & \text{for } r' < 0 \iff \text{sgn } dt \neq \text{sgn } r \\ \text{off}_B - r' = \text{off}_B - \text{sgn } dt * r & \text{for } r' > 0 \iff \text{sgn } dt = \text{sgn } r \end{cases} \\
 \text{thr}_R &= |r'| \\
 & \text{(where } r = \text{intv} - |dt| * n_g)
 \end{aligned}$$

(note that because  $\text{thr}_R = |dt_R|$ , recursive cycles are not affected by length drift - their check range will always have a length of 1)

All of this implies that the cycle indices  $c$  during which spinners change groups are also cyclic, with spinners changing groups ever group cycle index  $c$  where  $c = j * n_{g_R} + g_R$ , where  $n_{g_R} = \left\lfloor \frac{\text{intv}_R}{|dt_R|} \right\rfloor$  is the period of these recursive “group drift cycles”, and  $g_R$  is the recursive “group drift cycle group”. **When  $r < 0$ , group drifts decrement the group index** (= load checks happen one frame earlier than expected), **and when  $r > 0$ , they increment it** (= load checks happen one frame later than expected).

## Length drifts

As we derived above, the actual group of a spinner is not a discrete value, but instead a range of values, namely:

$$\begin{aligned}
 i &\in \begin{cases} [g; g + L) & \text{for } dt > 0 \\ (g - L; g] & \text{for } dt < 0 \end{cases} \\
 & \text{(where } L = \left\lceil \frac{\text{thr} - (-\text{off}_B \bmod |dt|)}{|dt|} \right\rceil)
 \end{aligned}$$

(from now on, we will consider  $L$  over  $off_c$  instead of  $off_B$ ) **TODO: Can this be safely done?**

$L$  is the length of this check range, and by setting  $thr = k * |dt| + o$ , where  $k = \lfloor \frac{thr}{|dt|} \rfloor$  (called the *base length*) and  $o = thr \bmod |dt|$  (called the *length offset*), we can reformulate  $L$  as

$$L = \left\lceil \frac{k * |dt| + o - (-off_c \bmod |dt|)}{|dt|} \right\rceil = k + \left\lceil \frac{o - (-off_c \bmod |dt|)}{|dt|} \right\rceil = k + l$$

Because of  $0 \leq o < |dt|$ ,  $l$  can only ever be 0 or 1, as  $-|dt| < o - (-off \bmod |dt|) < |dt|$ .  $l$  is 1 (which is called a *length drift*) only if

$$\begin{aligned} & o - (-off_c \bmod |dt|) > 0 \\ \Leftrightarrow & o > (-off_c \bmod |dt|) \\ \Leftrightarrow & -off_c < o \bmod |dt| \\ \Leftrightarrow & -(off_B + r' * c) < o \bmod |dt| \\ \Leftrightarrow & \mathbf{c} * (-\mathbf{r}') - off_B < \mathbf{o} \bmod |dt| \end{aligned}$$

This defines another recursive cycle with

$$\begin{aligned} dt_L &= -r' \\ t_L &= c \\ inv_L &= |dt| \\ off_L &= off_B \\ thr_L &= o = thr \bmod |dt| \end{aligned}$$

**Note that this almost exactly matches the definition of the regular recursive cycle!** The only two differences are:

- $|r'| = thr_R \neq thr_L = o$ 
  - This difference does not affect the group indices  $g_R = g_L$ , however a further recursive cycle is required to predict the recursive length drift of  $L_L$ .
- when  $r' > 0$ :  $off - r' = off_R \neq off_L = off$ 
  - In this case, the recursive group drift cycle is one tick behind of the length drift cycle. The regular recursive cycle can still be used to predict the length drift, but one must look at the values from the next tick instead of the current value. The recursive length drift cycle of  $L_L$  must then also be adjusted to use  $off_R$  instead of  $off_L$ , as this cancels out the fact that values are taken from the next tick.
  - This also means that length drifts either happen in the same group cycle, or in the group cycle before an actual group drift happens.

**TODO: Length drift inversion for potentially cleaner results?**

## Why this even happens

Careful readers might have noticed that all of the above analysis implicitly assumed that  $r = \text{intv} \bmod dt \neq 0$ , meaning that spinner offsets drift at all. However, as  $dt = \frac{1}{60}$  and  $\text{intv} = 0.05$ ,  $r$  should be  $r = 0.05 \bmod \frac{1}{60} = 0$ . Then why is the above analysis relevant at all?

Simply put, in practice, the imprecisions of floating point numbers mean that neither  $dt$  nor  $\text{intv}$  are their ideal, non-drifting values. The following effects cause them to be ever so slightly different:

- 0.05 is not exactly representable as a floating point number - the values used by the game's calculations is actually 0.0500000007450580596923828125 (this also affects  $dt$ )
- **TimeActive** is incremented every frame by 0.016666699200868606567382812500 (closest float value of 0.0166667). However, as it only is a 32 bit float, depending on the current magnitude of **TimeActive**, the actual *effective*  $dt$  will vary slightly because of float cancellation, and become less and less precise as time goes on. This results in “ranges” of different effective  $dt$  values, which will be examined later.
  - Note that *thr* is unaffected this effective deltatime imprecision - it will always remain the at the same precision and always equal exactly 0.016666699200868606567382812500

## The effective deltatime ranges

**TimeActive** is stored in a 32 bit IEEE754 normalized floating point number. As such, it is comprised of a 23 bit mantissa  $m = 1.\text{XXXXXXXXXXXXXXXXXXXXX}$  (in binary), an 8 bit exponent  $e = \text{XXXXXXXX}$  and a sign bit  $s$ , which will always be zero. The value of such a float is given by  $m * 2^{e-127}$ .

When adding two floating point numbers (like **TimeActive** and **DeltaTime**) with different exponents, a phenomenon called “float cancellation” occurs. This means that the lowest bits of the mantissa, which can now no longer be stored in the result are cut off. Additionally, the mantissa is rounded either up or down, depending on the highest trimmed of bit.

All of this effectively causes the effective deltatime to loose precision as the exponent of **TimeActive** gets bigger and bigger. Note that the effective deltatime is the same between two values of **TimeActive** with the same exponent, which allows for entire ranges of **TimeActive** values to be assigned the same effective deltatime. One edge case has to be taken into account though, which is when the next **TimeActive** value is already in the next range. In this case, the effective deltatime will be unique in some cases for one frame.

Included in the appendix are tables with various bits of information regarding the effective deltatime ranges.



## Appendix: Some data

### Effective Deltatime - Frame Ranges

TimeActive Exponent	Start Frame	End Frame
121	00000000	00000001
122	00000001	00000003
123	00000003	00000007
124	00000007	00000014
125	00000014	00000029
126	00000029	00000059
127	00000059	00000119
128	00000119	00000240
129	00000240	00000479
130	00000479	00000960
131	00000960	00001920
132	00001920	00003840
133	00003840	00007679
134	00007679	00015361
135	00015361	00030724
136	00030724	00061452
137	00061452	00122683
138	00122683	00246044
139	00246044	00492768
140	00492768	00986216
141	00986216	01918283
142	01918283	04015435
143	04015435	08209739
144	08209739	16598346
145	16598346	24986954
146	24986954	...

### Effective Deltatime - First TimeActive value in range

TimeActive Exponent	First TimeActive value in range
121	0.016666699200868606567382812500
122	0.033333398401737213134765625000
123	0.066666796803474426269531250000
124	0.133333608508110046386718750000
125	0.250000476837158203125000000000
126	0.500001132488250732421875000000
127	1.000002503395080566406250000000
128	2.000001668930053710937500000000

TimeActive Exponent	First TimeActive value in range
129	4.016666412353515625000000000000
130	8.000053405761718750000000000000
131	16.016597747802734375000000000000
132	32.016353607177734375000000000000
133	64.015869140625000000000000000000
134	128.012878417968750000000000000000
135	256.014953613281250000000000000000
136	512.002441406250000000000000000000
137	1024.010742187500000000000000000000
138	2048.015625000000000000000000000000
139	4096.000976562500000000000000000000
140	8192.004882812500000000000000000000
141	16384.013671875000000000000000000000
142	32768.003906250000000000000000000000
143	65536.007812500000000000000000000000
144	131072.015625000000000000000000000000
145	262144.000000000000000000000000000000
146	524288.000000000000000000000000000000

### Effective Deltatime - Effective $dt$ values

TimeActive Exponent	Effective $dt$ value
121	-
122	0.0166666992008686065673828125
123	0.0166666954755783081054687500
124	0.0166666954755783081054687500
125	0.0166667103767395019531250000
126	0.0166667103767395019531250000
127	0.0166666507720947265625000000
128	0.0166666507720947265625000000
129	0.0166668891906738281250000000
130	0.0166664123535156250000000000
131	0.0166664123535156250000000000
132	0.0166664123535156250000000000
133	0.0166702270507812500000000000
134	0.0166625976562500000000000000
135	0.0166625976562500000000000000
136	0.0166625976562500000000000000
137	0.0167236328125000000000000000
138	0.0166015625000000000000000000
139	0.0166015625000000000000000000
140	0.0166015625000000000000000000

TimeActive Exponent	Effective $dt$ value
141	0.017578125000000000000000000000
142	0.015625000000000000000000000000
143	0.015625000000000000000000000000
144	0.015625000000000000000000000000
145	0.031250000000000000000000000000
146	0.000000000000000000000000000000

### Effective Deltatime - Transition $dt$ values

TimeActive Exponent	Transition $dt$ value
121	0.0166666992008686065673828125000000000000
122	0.0166666992008686065673828125000000000000
123	0.0166666954755783081054687500000000000000
124	0.0166666954755783081054687500000000000000
125	0.0166666805744171142578125000000000000000
126	0.0166667103767395019531250000000000000000
127	0.0166666507720947265625000000000000000000
128	0.0166668891906738281250000000000000000000
129	0.0166664123535156250000000000000000000000
130	0.0166673660278320312500000000000000000000
131	0.0166664123535156250000000000000000000000
132	0.0166702270507812500000000000000000000000
133	0.0166625976562500000000000000000000000000
134	0.0166778564453125000000000000000000000000
135	0.0166625976562500000000000000000000000000
136	0.0166625976562500000000000000000000000000
137	0.0166015625000000000000000000000000000000
138	0.0168457031250000000000000000000000000000
139	0.0166015625000000000000000000000000000000
140	0.0175781250000000000000000000000000000000
141	0.0175781250000000000000000000000000000000
142	0.0195312500000000000000000000000000000000
143	0.0234375000000000000000000000000000000000
144	0.0156250000000000000000000000000000000000
145	0.0312500000000000000000000000000000000000
146	-

### Recursive cycle periods

Note: cycle 0 is the spinner group cycle (which always has a length of 3 for all exponents other than 145), and cycle 1 is the first recursive group change cycle.

TimeActive	Exponent	C0	C1	C2	C3	C4	C5	C6	C7	C8
121		0	0	0	0	0	0	0	0	0
122		3	172074	3	9	0	0	0	0	0
123		3	194519	5	2	2	0	0	0	0
124		3	194519	5	2	2	0	0	0	0
125		3	127827	4	9	0	0	0	0	0
126		3	127827	4	9	0	0	0	0	0
127		3	344148	3	4	0	0	0	0	0
128		3	344148	3	4	0	0	0	0	0
129		3	24994	3	12	2	2	0	0	0
130		3	21824	3	5	13	0	0	0	0
131		3	21824	3	5	13	0	0	0	0
132		3	21824	3	5	13	0	0	0	0
133		3	1561	6	3	10	4	2	2	0
134		3	1365	12	273	0	0	0	0	0
135		3	1365	12	273	0	0	0	0	0
136		3	1365	12	273	0	0	0	0	0
137		3	98	7	48	3	4	6	2	0
138		3	85	3084	17	0	0	0	0	0
139		3	85	3084	17	0	0	0	0	0
140		3	85	3084	17	0	0	0	0	0
141		3	6	2	3	11651	2	4	0	0
142		3	5	838861	0	0	0	0	0	0
143		3	5	838861	0	0	0	0	0	0
144		3	5	838861	0	0	0	0	0	0
145		2	3	2	1677721	0	0	0	0	0
146		0	0	0	0	0	0	0	0	0

How this can be used to efficiently predict spinner group changes

TODO