Celeste Spinner Analyisis - the SPIN theory

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:

$$n_g = \lfloor \frac{interval}{dt} \rceil = \lfloor interval * 60 \rceil$$

Note that the group count can be chosen arbitrarily as either $\lfloor \frac{interval}{dt} \rfloor$ or $\lceil \frac{interval}{dt} \rceil$ (except for $\frac{interval}{dt} \in (1;2)$, in which case one has to round upwards), though the bigger $|interval - n_g * dt| = |r|$ is, where $r = interval - n_g * dt$ is called the *residual drift* of the groups, the less the spinners will follow the group pattern consistently over time.

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):

$$g_{off} = \lceil \frac{off}{dt} \rceil = \lceil off * 60 \rceil \mod n_g$$

*Note: the above equivalence $\lceil \frac{off}{dt} \rceil = \lceil off * 60 \rceil$ assumes that dt is exactly $\frac{1}{60}$-th of a second. The actual dt obersved in game will be differemt vary over time, which will be explored later.

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 strictily 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 mathmatical notation yields

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

which is equivalent to

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

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

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

For purposes which will become apparent later, let us relax our assumptions slightly from our implict $dt > 0$ 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. The relaxed statement we will continue to further analyse is:

$$t * dt - off < |dt| \mod intv$$

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:

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

Let us now define the *residual drift* r as $r = intv - |dt| * n_g$ just like we did earlier. Inserting this new symbol into our condition we get:

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

Note that the final expression differs from what we would expect if all spinner cycles were exactly equal! Instead of $i * dt - off < |dt| \mod intv$, 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 $i * dt - (off + \operatorname{sgn} dt * r * c) < |dt| \mod intv$. **This means that the effective offset of each spinner shifts by** $\operatorname{sgn} dt * r$ **every** n_g **frames!**

When group changes happen

A spinner will change its group as soon as $i * dt - (off + \operatorname{sgn} dt * r * c) < |dt| \mod intv$ no longer evaluates to the same value as $i * dt - off < |dt| \mod intv$. For convience, let $r' = \operatorname{sgn} dt * r$ and $off_c = off + r' * c$, . The two expressions from above can now be written as $i * dt - off < |dt| \mod intv$ and $i * dt - off_c < |dt| \mod intv$.

Observe that the expression, $i * dt - off < |dt| \mod intv$ is only ever true for a single value of i, namely g, the spinner's group index. Solving for g yields $g = \operatorname{sgn} dt * \lceil \frac{off}{|dt|} \rceil \mod n_g$. Note that we will use a slightly different definition of g for the rest of this analysis, namely $g' = \lceil \frac{off}{|dt|} \rceil$ which is lacking the $\operatorname{sgn} dt$

term. This definition is useful as it still allows for checking if group changes occur, but also has the following relation hold:

$$r' > 0 \iff \mathit{off}_{c-1} < \mathit{off}_c \iff g'_{c-1} \leq g'_c$$

We can now differentiate two different cases depending on the sign of r': - When $r' < 0$, then $g'_{c-1} \geq g'_c$. As such we want to find all values of c for which $g'_{c-1} > g'_c \iff \lceil \frac{\mathit{off}_{c-1}}{|dt|} \rceil > \lceil \frac{\mathit{off}_c}{|dt|} \rceil$ holds. Multiplying both sides by -1 we get:

$$\begin{aligned} -\lceil \frac{\mathit{off}_{c-1}}{|dt|} \rceil &< -\lceil \frac{\mathit{off}_c}{|dt|} \rceil \\ \iff \lfloor \frac{-(\mathit{off} + c * r' - r')}{|dt|} \rfloor &< \lfloor \frac{-(\mathit{off} + c * r')}{|dt|} \rfloor \\ \iff \lfloor \frac{\mathbf{c} * (-\mathbf{r}') - \mathit{off} - (-\mathbf{r}'))}{|dt|} \rfloor &< \lfloor \frac{\mathbf{c} * (-\mathbf{r}') - \mathit{off}}{|dt|} \rfloor \end{aligned}$$

- When $r' > 0$, then $g_{c-1} \leq g_c$. As such we want to find all values of c for which $g_{c-1} < g_c \iff \lceil \frac{\mathit{off}_{c-1}}{|dt|} \rceil < \lceil \frac{\mathit{off}_c}{|dt|} \rceil \iff \lceil \frac{\mathit{off}_{c-1}}{|dt|} \rceil < \frac{\mathit{off}_c}{|dt|}$ holds. Multiplying both sides by $|dt|$ and substituting $\lceil \frac{a}{b} \rceil * b = a + (-a \bmod b)$ we get:

$$\begin{aligned} \mathit{off}_{c-1} + (-\mathit{off}_{c-1} \bmod |dt|) &< \mathit{off}_c \\ \iff \mathit{off}_{c-1} + (-\mathit{off}_{c-1} \bmod |dt|) &< \mathit{off}_{c-1} + r' \\ \iff -\mathit{off}_{c-1} \bmod |dt| &< r' \end{aligned}$$

Because $0 < r' < dt$ (as $|r'| = |\mathit{intv} - n_g * |dt|| = |\mathit{intv} - \lfloor \frac{\mathit{intv}}{|dt|} \rceil * |dt|| < |dt|$), it holds that $r' \bmod |dt| = r'$, and as such:

$$\begin{aligned} \iff -\mathit{off}_{c-1} &< r' \pmod{|dt|} \\ \iff -(\mathit{off} + r' * c - r') &< r' \pmod{|dt|} \\ \iff \mathbf{c} * (-\mathbf{r}') - (\mathit{off} - \mathbf{r}') &< |\mathbf{r}'| \pmod{|dt|} \end{aligned}$$

- When $r' = 0$, then there is no drift - the recursive cycle sequence ends here

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 indices of spinner cycles when spinners change groups behave just like a recursive instance of the original problem with the following parameters:

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

This means 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 $n_{g_R} = \lfloor \frac{intv_R}{|dt_R|} \rceil$ is the period of these recursive "group drift cycles", and g_R is the recursive "group drift cycle group", which initially starts as $g_R = \lceil \frac{off_R}{|dt_R|} \rceil \bmod n_{g_R}$, **but is then also recursively affected by group drifting!**

Why this even happens

Careful readers might have noticed that all of the above analysis implicitly assumed that $r = intv \bmod dt \neq 0$, meaning that spinner offsets drift at all. However, as $dt = \frac{1}{60}$ and $intv = 0.05$, 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 *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 $\frac{1}{60}$-ish. 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.

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.XXXXXXXXXXXXXXXXXXXXXXX` (in binary), an 8 bit exponent `e = 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.

The following tables contain of various bits of information regarding the effective deltatime ranges:

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	...

First `TimeActive` value in range

`TimeActive` Exponent	First `TimeActive` value in range
121	0.0166666992008686065673828125000
122	0.0333333984017372131347656250000
123	0.0666667968034744262695312500000
124	0.1333336085081100463867187500000

`TimeActive` Exponent	First `TimeActive` value in range
125	0.25000047683715820312500000000
126	0.50000113248825073242187500000
127	1.00000250339508056640625000000
128	2.00000166893005371093750000000
129	4.01666641235351562500000000000
130	8.00005340576171875000000000000
131	16.0165977478027343750000000000
132	32.0163536071777343750000000000
133	64.0158691406250000000000000000
134	128.012878417968750000000000000
135	256.014953613281250000000000000
136	512.002441406250000000000000000
137	1024.01074218750000000000000000
138	2048.01562500000000000000000000
139	4096.00097656250000000000000000
140	8192.00488281250000000000000000
141	16384.0136718750000000000000000
142	32768.0039062500000000000000000
143	65536.0078125000000000000000000
144	131072.015625000000000000000000
145	262144.000000000000000000000000
146	524288.000000000000000000000000

Effective `DeltaTime` values

`TimeActive` Exponent	Effective `DeltaTime`
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

TimeActive Exponent	Effective DeltaTime
137	0.01672363281250000000000000000
138	0.01660156250000000000000000000
139	0.01660156250000000000000000000
140	0.01660156250000000000000000000
141	0.01757812500000000000000000000
142	0.01562500000000000000000000000
143	0.01562500000000000000000000000
144	0.01562500000000000000000000000
145	0.03125000000000000000000000000
146	0.00000000000000000000000000000

Transition DeltaTime values

TimeActive Exponent	Transition DeltaTime
121	0.01666669920086860656738281250000000000000
122	0.01666669920086860656738281250000000000000
123	0.01666669547557830810546875000000000000000
124	0.01666669547557830810546875000000000000000
125	0.01666668057441711425781250000000000000000
126	0.01666671037673950195312500000000000000000
127	0.01666665077209472656250000000000000000000
128	0.01666688919067382812500000000000000000000
129	0.01666641235351562500000000000000000000000
130	0.01666736602783203125000000000000000000000
131	0.01666641235351562500000000000000000000000
132	0.01667022705078125000000000000000000000000
133	0.01666259765625000000000000000000000000000
134	0.01667785644531250000000000000000000000000
135	0.01666259765625000000000000000000000000000
136	0.01666259765625000000000000000000000000000
137	0.01660156250000000000000000000000000000000
138	0.01684570312500000000000000000000000000000
139	0.01660156250000000000000000000000000000000
140	0.01757812500000000000000000000000000000000
141	0.01757812500000000000000000000000000000000
142	0.01953125000000000000000000000000000000000
143	0.02343750000000000000000000000000000000000
144	0.01562500000000000000000000000000000000000
145	0.03125000000000000000000000000000000000000
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 efficently predict spinner group changes

TODO