Not quite *One in a million*: The probability behind Chuya Takizawa's Attack Ship 2:06

Whiteted, whiteted.strats@gmail.com

April, 2018
Version 2

1 Introduction

This article and the attached C# files are a response to RWhiteGoose's video on the topic, titled *The "Luckiest" Speedrun in Perfect Dark History: Real or Fake? [Complete Analysis]* [`https://www.youtube.com/watch?v=cJMQDY8Ci90`]. This is the 2nd version after large corrections taken on board from the related forum thread (`https://forums.the-elite.net/index.php?topic=22418.0`).

Section 2 boils up Perfect Dark's Attack Ship stage to give an accurate mathematical model, so as to answer 2 questions:

1. What are the chances of such a 2:06 pace ending?
2. What are the chances of a 2:08 pace ending? (Big Bossman's run)

With quite conservative values for the model's parameters, discussed in Section 2.3, the attached C# program gives the following results:

1. 1 in 119
2. 1 in 12

The final section discusses the significance of these values and gives recommendations. We omit a lot of chat from the first version, since the probabilities here are a lot higher (more likely) so they largely speak for themselves.

2 The model

2.1 Ignored subtleties

The model makes the following assumptions:

1. Frame rate & lag
   The game runs consistently at 20fps, making all decisions on each frame, in particular whether to spawn an enemy. The game actually runs at 60fps, but such decisions seem to be made approximately once every 3 frames.
2. Human error
   The player follows the given strategy perfectly. This is actually quite reasonable as we shall see.
3. Non-randomness in the RNG
   The RNG (random number generator) is perfectly random. This is in fact a reasonable assumption since controller input, which can be modelled as slightly random, is fed into the RNG. It is important to realise what that this argument suggests that the first 'roll' of the RNG in the ending is 'truely random'. It does not however suggest that subsequent rolls will be random: The main non-randomness of RNGs is in their lack of independence between closely timed rolls.
4. Despawns irrelevant
   We understand that the game considers enemies to be dead at the start of the death animation. Unlike in Goldeneye, the death animation is not relevant.

2.2 The game: definition and strategy

The game together with our player's simple strategy is set out in Algorithm 1 below. The game starts at the what RWhiteGoose describes as the earliest point that Skedars #1 and #3 can uncloak. We await the spawn then kill the first Skedar, move to the right door, then await the spawn and kill the second two Skedars. For Joanna to kill Skedar #2 it must first open the right door. See figure 1 and algorithm 1 below.

The 'game' we consider has 4 parameters, though only the first is truely considered variable:

t_{target}	The time limit in frames*
t_{move}	The number of frames* it takes the player to move in step 3
t_{open}	The number of frames* it takes Skedar #2 to open the door.†
t_{kill}	The number of frames* to react to a spawn

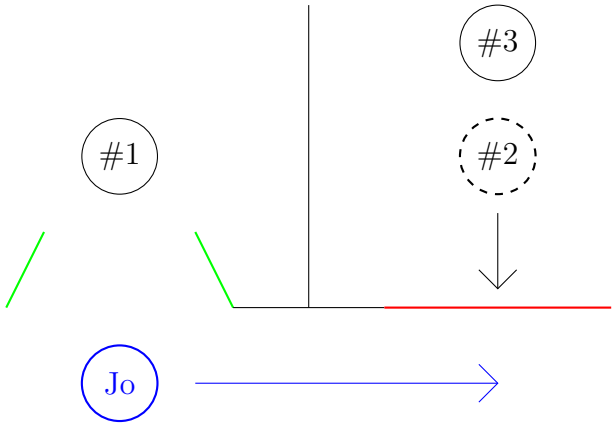


Figure 1: Game picture

Algorithm 1 Game & Player strategy

1: Await spawn #1
2: Kill Skedar #1
3: Move to right door
4: Await spawn #2
5: Await Skedar #2 opening door
6: Kill Skedar #2
7: Await spawn #3
8: Kill Skedar #3

* Recall that our game operates at 20fps.
† This may actually be somewhat random, coming from the RNG. We consider it constant for now.

Also note that all of the above could be replaced by some probability distributions for a better model, but for simplicity we'll keep them constant. We may also consider lag and add a lag factor as a parameter in the future.

The game is considered a success if we manage to kill all 3 enemies inside the time limit t_{target}.

2.3 The game: parameter values

These values are set in the attached C# script.

$$\begin{aligned} t_{move} &= 28 \\ t_{open} &= 70 \\ t_{kill} &= 8 \end{aligned}$$

$t_{move} = 28$ or 1.4s is taken from Big Bossman's 2:08 run, where he kills Skedar #1 at approximately 1:59.70, and camps at the right door at 2:01.13. $t_{open} = 70$ or 3.5s is taken from Henrik's facts topic (`https://forums.the-elite.net/index.php?topic=19067.msg439221#msg439221`) and $t_{kill} = 8$ is chosen without much thought. The first two of these are probably slightly large, waiting until Joanna was completely stationary at the right door and adding .2s to Henrik's 3.3s value. The last is hard to estimate - for example I am not clear if it needs to include any time for the Skedar to decloak. Nonetheless 8 frames should be on the side of caution.

The only remaining parameter is t_{target}. In the 2:06 run, RWhiteGoose estimates that the ending starts at 1:58.60. Skedar #3 appears to be killed at 2:04.60, giving $t_{target} = 120$, or 6s. Similarly, the 2:08 seems to have a 1:57.95 ending start, and ends at approximately 2:06.20, giving $t_{target} = 165$. There are certainly some errors here which could be improved - the 2:08's timer is hidden by ammo messages in most of the ending.

Run	t_{target}
2:06	120
2:08	165

2.4 The game: spawn mechanics

Each of the left and right rooms are completely independent. This was not understood in the first version of this article. Skedar's #1 and #3 start a timer lasting 5 seconds. Each frame this has a 1 in 256 chance to finish early (except the 100th frame, where it isn't early). When the timer expires, an *event* is triggered, and the timer resets to 5s (provided there are more events to trigger).

Skedar #1 has a single event: to decloak and become vulnerable to standard damage. Skedar #3 has two, provided the player follows the correct strategy. The first is to spawn Skedar #2, and the second is to decloak as #1 does. Crucially the second will be 'spuriously triggered' if Skedar #2 is not killed. That is to say suppose Skedar #2 has been spawned but not killed, when Skedar #3's decloak event is triggered. The timer *will* be reset, but Skedar #3 will not decloak. The decloak event will be 'rescheduled' for the next timer loop.

Let us make an assumption explicit: Skedar #2 doesn't get much beyond the door, even if it spawns on frame 1 and Skedar #1 doesn't uncloak for nearly the full 5 seconds.

Recalling that Skedar #2 takes 3.3 seconds to crack the door, this seems reasonable. Then the door must open fully and Skedar #2 must navigate

towards Joanna. As a result we further model Skedar #2 as not getting past the door. This is because even if it does, the player should not kill it until they have moved Joanna across to the door, in view of Skedar #3. This is to prevent Skedar #3 from spawning another Skedar.

3 The mathematics

3.1 RWhiteGoose's approaches

RWhiteGoose attempts to use some very clean mathematics in his 2nd approach of the video by asking what are the chances of drawing atleast 3 'successes' (spawns) in N trials, where for the 2:06 run $N = 35$. This has two issues. Firstly it does not consider that the 2nd enemy may spawn while the player is travelling away from the 1st kill. RWhiteGoose does address this later in the video, but we won't discuss that here. The more important issue is that it does not consider the possibility of a timeout. This causes large error when considering the 2:08 or 2:09 pace endings, where a successful run is more likely to utilise a timeout.

Moreover, RWhiteGoose's video essentially asks what the probability of that specific time is, i.e. $Pr(t_{target} = 120)$. A fairer statistic to establish the run's luck is $Pr(t_{target} \leq 120)$.

3.2 Markov chain introduction

A not so clean but very practical approach is a Markov Chain.

Definition 1. *A Markov chain*

A Markov chain consists of a series of states, or nodes. Each one has arrows to some of the other states. Each arrow is marked with a probability. We have a start state and some target or success state(s).

To understand Markov chains you should imagine standing on one. You start on the start state. The probabilities on the arrows pointing away from you represent the chance that you will walk along those arrows. For instance if there are 2 arrows each with probability 1/2, you toss a coin and if heads walk along the left edge, if tails the right edge. We can use them to ask the burning question, "what is the chance of reaching a target state from the start state on any given walk?".

3.3 Markov chain approach: States

We make an extensive list of what values we need to sufficiently capture the current *state* of the game:

1. Time elapsed / remaining
2. Left timer (Skedar #1)
3. Right timer (Skedar #3)
4. Stage - i.e. "moving to right door"
5. Killing timer (all Skedars)
6. Time since Skedar #2 spawn
7. Time since we started moving to the door

Now we reduce redundancy, and determine the precise set of stages. Observe that since the left timer only has one event, it is never reset. Hence it can be deduced from stage and time elapsed, where stage and kill timer distinguish the first spawn. Conversely since the right timer could spawn Skedar #2 and then finish early many times before it is killed, the right timer is essential.

Now, with some thought we can also combine the last two timers in our list, into the M (movement) timer. We also introduce two stages: {Joanna and Skedar #2 moving} and {Just Joanna moving}. If the stage is before this point then Joanna is not moving. If *and only if* the timer is -1 then neither are moving. This captures all cases very efficiently, as even just introducing a bit for { Skedar #2 spawned } essentially doubles the possible number of states. As the 2nd of these two starts moving, the timer is adjusted to be the maximum of what is left and the new actor's movement time.

So our state consists of:

$$(t_{elap},\, T_R,\, t_M,\, t_K,\, stage)$$

t_{elap}	Time elapsed	0	to	≤ 300
T_R	R (Right) timer	0	to	100
t_M	M timer	-1	to	≤ 70
t_K	K (Killing) timer	0	to	t_{kill}
$stage$	Stage (detail below)	a	to	d

Where our 4 stages are:

a. Awaiting / Killing Skedar #1 *
b. Just Joanna moving
c. Both Joanna and Skedar #2 moving.. or killing Skedar #2 *†
d. Awaiting / Killing Skedar #3 *

* Notice that whether we are killing or awaiting can be deduced from the K timer value: If it is 0 we must be awaiting, otherwise we'd be on to a later stage.

† Moreover in stage c. this can be combined with testing if the M timer has expired (is 0) so see that we've both stopped moving.

The start state is (0, 100, -1, 0, a).

A first choice for the success states for a given t_{target} would be those of the form (t_{target}, 0, 0, 0, d), with the idea that the kill timer has recently hit 0 in stage d. But this is actually Awaiting Skedar #3. Instead we exploit a little leftover redundancy to make up a nonsense state as a final stage:

$$(t,\ 0,\ 1,\ 0,\ \mathrm{d}),$$

with transitions incrementing t. Notice this says Joanna or Skedar #2 still need to move, despite us being at Skedar #3.

3.4 Practical programming considerations

Conservatively saying $t_{kill} = 16$, this gives us a limit of atmost

$$301 * 101 * 72 * 9 * 4 = 78,799,392$$

possible states. That's pretty tight to what is plausible but we can just about process it. We will probably move to C# (from Python3) to cope.

In my original thinking I was going to compute the entire Markov chain with labelled transitions first, and then ask questions of it. This led me to hold the entire structure in memory, which has persisted despite abandoning this idea. As a result, we compress these states into a single integer, to reduce the memory required.

The final algorithm exploits that all transitions increment $t_{elapsed}$ and so the Markov chain is a directed acyclic graph. We essentially perform a breadth first search, determining the probabilities of reaching states with lower $t_{elapsed}$ values earlier.

All details and extensive comments are in the C# file(s).

3.5 Transitions

Recall states are of the form

$$(t_{elap},\ T_R,\ t_M,\ t_K,\ stage)$$

The transitions look reasonable complex but essentially capture two ideas:

1. Determininistic events
   These are events which do not involve probability. For example whenever $t_M > 0$, it is decremented by 1 in a transition (except in the success state).

2. Probabilistic events
   These are only the timers. For example considering just the Right timer when $t_R > 0$ we have two transitions:

$$(t,\, r,\, ...) \rightarrow (t+1,\, r-1,\, ...)$$
$$(t,\, r,\, ...) \rightarrow (t+1,\, 0,\, ...)$$

   with probabilities 255/256 and 1/256 respectively. Notice I have omitted the later aspects of the state for simplicity: t_R hitting 0 may trigger the state to change, and t_K too. In the code multiple deterministic events and probabilistic events are often rolled into a single transition.

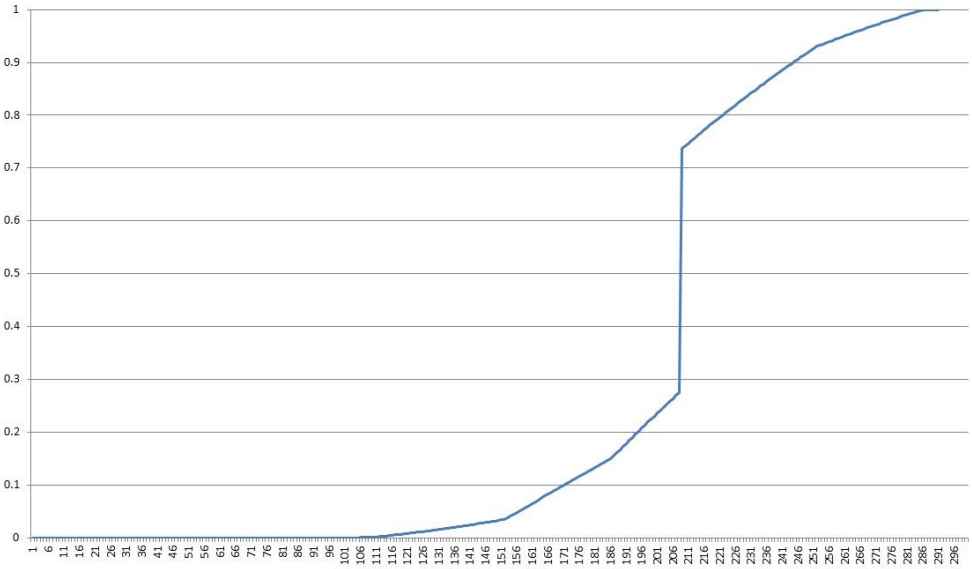
We leave the remaining details to the C# file, Program.cs.

3.6 Markov chain results

Taking the values from section 2.3, our program yields the probabilities that we claimed in the introduction:

Run	t_{target}	Pr. 1 in _
2:06	120	119
2:08	165	12.15

The entire cumulative distribution function looks quite odd:



We name these almost linear sections from the left:

1. The floor: 0 to 109, where the function is almost 0.
2. The toes: 110 to 152
3. The foot: 153 to 186
4. The ankle: 187 to 208
5. The shin: At 208, the leap up
6. The knee: 209 to 252
7. The rest!

The raw distribution can be found in an attached csv file (or as output from the C# file).

4 Conclusion & Recommendations

We have effectively analysed the probability involved in the ending of the stage Attack Ship. An ending atleast as good occurs with probability 1 in 119 under reasonable assumptions, and this is with slightly pessimistic or overlarge values for the game's parameters. Concretely we can say that the 2:06 run was not in fact (music plays) 1 in a million, provided that fast starts are not extremely rare.

Analysing how hard this time would be to tie really needs data on the distribution of starts, since an even faster ending, though very unlikely, could contribute a significant amount of probability if slightly slower starts were comparatively very likely. In the last version I suggested that the run was 1 in 10,000. Without data on the starts I don't want to drop this figure but

obviously I am more confident it is true. I will happily perform the analysis if someone gives me the data.

As a final thought all of this is built on the current understanding of the level's mechanics. Great thanks to Henrik for clearing up a lot of my misunderstandings there. The new model captures 'reality' much more closely. The only aspect which I am unsure on is if the Skedars have some 'invincibility frames' while they are decloaking. This can still fit in the current model by adjusting the value of t_{kill}, and indeed at 8 it may already be large enough.