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

Whiteted, whiteted.strats@gmail.com

April, 2018

1 Introduction

This article and the accompanying Python scripts are a response to RWhite-Goose'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`].

We boil up RWhiteGoose's video to give a reasonable mathematical model of the ending to Perfect Dark's Attack Ship stage, and answer 3 non-technical 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)
3. How do these compare?

We will defend quite rigorously the following answers:

1. 1 in 1037
2. 1 in 133
3. The 2:06 *ending* is at most "8 times as lucky"

In particular we will explain what the 3rd question and answer even mean. Additionally we will try to seat some ideas in a wider context and explain some of the common statistical pitfalls which this problem raises, and which have made themselves into RWhiteGoose's video. We will even be brave and delve into the youtube comments to answer some points raised. RWhiteGoose's video was incredibly interesting and entertaining and I would like to thank him for it.

2 The model

2.1 Ignored subtleties

Firstly we discuss how we model the situation. Notably there are several things we ignore:

1. Frame rate & lag
   We consider that the game runs consistently at 10fps, and makes all decisions on each frame, in particular whether to spawn an enemy.
2. Human error
   We consider a player following the given strategy perfectly. This is actually quite reasonable as we shall see.
3. Non-randomness in the RNG
   We consider a perfect RNG. If we listen carefully we can hear "But Whiteted, the random number generator (RNG) is deterministic!" in our heads. It is a good exercise, which I will make a short video on sometime, to demonstrate that controller input is fed into Goldeneye's RNG. I assume it is the same for Perfect Dark. Modelling controller input as (slightly) random is more valid, and since the run is over 2 minutes in length, or 1200 frames, there is plenty of opportunity for randomness to enter 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. This is all quite technical and needs a proper study of the RNG, which I plan on doing sometime.
4. Fixed length despawns
   We assume the enemies despawn in a fixed amount of time, unlike in Goldeneye. This assumption is needed to fix the earliest spawn time of the second enemy relative to when the player starts moving.

2.2 The game: definition and strategy

We describe the game together with our player's simple strategy in Algorithm 1 below. The game starts at the what RWhiteGoose describes as the earliest point that the first enemy can spawn. We await then kill the first spawn, move, then await and kill the second two spawns.

The 'game' we consider has 2 parameters:

t_{target}	The time limit in frames
t_{move}	The number of frames it takes the player to move in step 3

Algorithm 1 Player strategy

```
1: Await spawn A
2: Kill enemy A
3: Move to spawn location B
4: Await spawn B
5: Kill enemy B
6: Await spawn C
7: Kill enemy C
```

We further define $t_{wait} = t_{target} - t_{move}$.

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

We take our values for t_{target} and t_{move} from RWhiteGoose's video. As reported, in his 2:08 Big Bossman kills the first spawn at 1:59.21 and reaches the door to the second spawn at 2:01.13.

$$t_{move} = 19$$

For the 2:06 run we have $t_{wait} = 16 + 6 + 13 = 35$. This figure comes from the "byte checks" referred to by Goose. Similarly for the 2:08 run we have $t_{wait} = 13 + 22 + 20 = 55$.

Run	t_{target}
2:06	54
2:08	74

2.4 The game: spawn mechanics & instakill

Each 'normal' frame where there is no enemy there is a 1 in 256 chance of the next enemy spawning. However if 49 frames pass without a spawn then on the 50th, or 'timeout', frame a spawn will be forced. We refer to this 50 frame figure as the timeout, T.

As a result, any events in the interval from the frame when an enemy spawns until the enemy is killed have no effect on the rest of the run. Hence we can assume that enemies are killed on the same frame they spawn, our 'instakill' assumption. If we want to adjust this, even with a probabilistic model of how long it takes a player to kill the enemy, we can do it after. Note that if the enemies take a random amount of time to despawn (as in

Goldeneye) this cannot be 'factored out', as the player may be moving during despawn, but we've assumed this is not the case.

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. We shall see that this only causes a small error for the 2:06 pace run (where a possible timeout spawn B and very fast spawns A,C are possible) but causes a very large error when considering the 2:08 or 2:10 pace endings, where a successful run is more likely to utilise a timeout.

3.2 Markov chain introduction

A not so clean but very practical approach is a Markov Chain. Googling this term to find a picture is quite helpful, but the essence is quite simple:

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

We take our states to be triples:

$$(n_{kills}, t_{rem}, T_{curr})$$

n_{kills}	Number of kills so far
t_{rem}	The remaining time left of t_{target}
T_{curr}	The number of frames until a spawn will be forced

Observe that this does capture everything necessary. To explain what this even means, consider removing the T_{curr} from our state: Our states no longer recognise how long until a spawn is forced. We cannot know if the probability of a spawn (and hence a kill via instakill) next frame is 1 or just 1/256.

The start state is (0, t_{target}, 50).

The success states are those of the form (3, 0, *any*).

The transitions (arrows) are a little more complex, but we describe them slightly loosely, with the details in the Python files:

1. Transitions where $T_{curr} = 0$
   Here a spawn is forced by the timeout, we have a single transition with probablity 1:

$$(\text{n}, t_{rem}, T_{curr}) \quad \rightarrow \quad (n+1, t_{rem} - 1, 50)$$

2. Transitions where $T_{curr} > 0$
   Here a spawn is not forced, but rather we have a 1/256 chance of a spawn, and so the above transition. The other transition has a 255/256 chance:

$$(\text{n}, t_{rem}, T_{curr})) \quad \rightarrow \quad (n, t_{rem} - 1, T_{curr} - 1)$$

3. Transitions where $n_{kills} = 0$
   This is a special case to capture the player's movement, and the chance of enemy B already being spawned. As above we have a 1/256 or 1 chance of a spawn, depending on the timeout. But we combine this with a $1 - (255/256)^{t_{move}}$ chance of the second spawn already being there.

   This yields two transitions:

$$\begin{array}{rcl} (0, t_{rem}, T_{curr})) & \rightarrow & (1, t_{rem} - t_{move} - 1, 50 - t_{move}) \\ & \rightarrow & (2, t_{rem} - t_{move} - 1, 50) \end{array}$$

   with probabilities that can be found in the Python file. Notice we leap forward more than 1 frame.

3.4 Markov chain results

Notice also that the each transition of our Markov chain decreases the time remaining. As a result we don't have any loops in the graph. The main brilliance of the theory of Markov chains involves loops, but fortunately for us our 'chain' forms a directed acyclic graph (DAG) and the analysis is a *lot* easier. Essentially we work back from the success states to mark each state with a probability of success from this point. This is all performed in *attack_ship.py*.

Taking the values from section 2.3 we get the probabilities that we claimed in the introduction:

Run	t_{target}	Pr. 1 in _
2:06	54	1037
2:08	74	133

4 Other Python files

attack_ship_sim.py uses the randint function from the random library to simulate playing the game which we have set out above, and roughly corroborates my results, albeit quite slowly.

attack_ship_no_timeout_approx.py changes the game so that there is no timeout. This allows a very clean analysis by conditioning on whether or not the second enemy spawns during the player's movement, which is set out in the file. We can compare the results with the timeout version:

t_{target}	Pr. 1 in _ with timeout	Pr. 1 in _ no timeout
50	1234	1234
54	1037	1079
60	523	677
70	167	418
74	133	375
80	66	279
90	37	197
100	24	145
200	1	23

Notice the small change for 54, the 2:06 value, and a large change for 74, the 2:08 value. This demonstrates that most (approximately 2/3) of endings with similar pace to Big Bossman's 2:08 would involve a timeout. We could make this precise and probably even say that the timeout would frequently be for the 2nd enemy, but we won't do that here.

Also the two agree on all values below 51, corroborating the correctness of my first Python file.

5 Statistics & wider comments

Statistics is a bit of a meme in mathematics. As we'll discuss in this section we must be very careful what questions we ask of the data, otherwise you can get whatever answer you seek.

5.1 Comparing the 2:06 and 2:08

Fix the duration of the start to be good pace, referred to by RWhiteGoose as a "1:47 Elvis". We ask "given such a run that is atleast as fast as the 2:08 run, what is the chance that it is atleast as fast as the 2:06 run?". We claim this is a fair way to compare the two.

We can compute this directly via Bayes theorem. Let A be the event that it is $\leq$ 2:08, and B that it is $\leq$ 2:10. Observe that P(B | A) = 1.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(A)}{P(B)}$$
$$P(A \mid B) = 12.81\%$$

which is just above 1 in 8, defending our claim from the introduction. This is a good statistic except that it doesn't consider the randomness in the start of a run. This would "benefit the 2:08 run" in the sense that there is a small chance of getting a 2:08 with a slower start (say a 1:48 Elvis), whereas a 2:06 would essentially require an optimal start. We would need to know the distribution of starts to make this analysis, but it could well say that the 2:06 is significantly more surprising than 1 in 8.

5.2 RWhiteGoose's 1st and 3rd approach errors

"Choice. The problem is choice." - Neo

RWhiteGoose's first approach demonstrates an incredibly common pitfall, which there is no shame in falling into. In the video he attempts to measure how surprising the 2:06 run is by computing the chance of those exact moments for a "byte check".

This is analogous to having someone roll a dice, then being surprised at whatever number comes up, because there was only "a 1/6 chance of that number occuring". The problem is choice. You chose the thing that was going to surprise you (a 5, or a 3, whatever was rolled) *after* the random event. It should not in fact surprise you at all.

A more valid approach would be to announce "I'll be surprised if it is a 6" before the roll. Then they roll it, and if it is a 6 you can be "1 in 6 surprised", whatever that means.

The 3rd approach, where RWhiteGoose applies 'leniency' unfortunately is no less invalid. You can immediately see that it fails to appreciate the possible combinations of number of byte checks taken just in the last two spawns.

5.3 Youtube comments!

Anonymised because it seemed sensible.

1. "Simply put there is not enough evidence to say whether is real or fake."

   We can make this more precise in the following sense. Suppose you accept the legitimacy of the 2:08. Then the 1 in 8 figure says you should not be very surprised by the luck in the 2:06 run, so that is certainly not enough to say that it is fake.

2. "You assume numbers to be random, but they are just an LFSR sequence. There's usually only tens thousands of possible sequences."

   I covered this reasonably comprehensively in section 2.1.

3. "What are the chances of getting 2:06 without first getting 2:09, 2:08 or 2:07? Surely that would be astronomical?"

   There are two points here. Firstly observe that the probability that this commenter is looking for is strongly related to the comparison that we made in section 5.1: Our figure is the chance that if you were getting atleast the 2:08 *ending* you would instead get atleast the 2:06 ending. So we can sort of say there was a 1 in 8 chance that Big Bossman would have skipped 2:08 and got 2:06 instead. So essentially no, it is not astronomical.

   Secondly this is very much choosing the question after the data.

4. "1/11007?.... 007..."

   A cute example of 'coincidences' occuring where you have choice over what 'surprises' you.

5.4 Choice of context

Ultimately, however you choose to analyse the probability you should then put it in a suitably wide context. Are we asking how surprised we are at the 'luck' in this particular stage, Attack ship? Or among all heavily luck-dependent stages of the game, Perfect Dark? Or among all similar stages of games that RWhiteGoose might do a video about? As you move into a wider context your surprise is diluted.

6 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 1037 under reasonable assumptions. Our crucial probability of 1/8 used in comparison with Big Bossman's 2:08 should be taken with the knowledge that it can only decrease as you consider the run as a whole rather than just the ending. More concretely we can say that provided a "1:47 Elvis" occurs more commonly than about 1 in 900 then the 2:06 run was not in fact (music plays) 1 in a million.

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. I will happily perform the analysis if someone gives me the data. For the sake of a conclusion I will claim this run was 1 in 10,000.

As a result I do not recommend that Karl Jobst attempt to tie this, but instead devote his time to Goldeneye (possible author bias here).

As a final thought all of this is built on the current understanding of the level's mechanics. I am not clear whether the earliest moment that the first enemy can spawn has been read out of the game's code or whether they have been deduced by many trials. It is plausible that they are slightly earlier than current thinking. Alternatively a major glitch speeding up the start would make this untied a lot weaker.

Okay time to eat some food now.