Home | Graphics | Documentation | Internals
Contributions | Downloads | Forum

WARNING !!!
WARNING !!!
WARNING !!!

IF YOU HAVE NOT PLAYED DUNGEONS OF DAGGORATH BEFORE
OR IF YOU HAVE NEVER BEATEN THE WIZARD
OR IF YOU SIMPLY WANT TO PRESERVE THE MYSTERY OF THE GAME
YOU SHOULD NOT READ ANY MORE OF THIS PAGE

WARNING !!!
WARNING !!!
WARNING !!!

THE INTERNALS PAGE IS INTENDED FOR PEOPLE WHO HAVE
PLAYED AND ESSENTIALLY MASTERED THE GAME AND WHO HAVE
BEEN WAITING UP TO TWENTY YEARS TO FIND OUT WHAT
IS REALLY GOING ON INTERNALLY IN THE GAME PROGRAM

WARNING !!!
WARNING !!!
WARNING !!!

SO IF YOU DON'T WANT TO SPOIL THE GAME FOR YOURSELF
THEN PLEASE DO NOT CONTINUE READING

WARNING !!!
WARNING !!!
WARNING !!!

YOU HAVE BEEN WARNED

WARNING !!!
WARNING !!!
WARNING !!!

==============================
==============================

```
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
=================================
```

Daggorath Internals

Introduction

Dungeons of Daggorath, unlike many of today's modern games, does not reveal any of its internal mathematical computations to the player. Some people like this about the game, and others dislike it. Personally, I think it makes the game more mysterious and therefore more fun to play. However, like many others who have played the game over the years and mastered (at least some of) its intricacies, there are many things I always wondered about regarding what exactly is going on behind the scenes.

The only visible and audible clue to the player's health is the heartbeat. This is engineered in such a way that it mimics a realistic heart rate, and most players find their own heart beating somewhat in tune with the game heart (though hopefully no one has actually fainted or died). The heartbeat is undoubtedly the most compelling aspect about the game that draws the player into the game world emotionally (and physiologically), and it certainly must have helped the game's initial sales (in comparison to other games of the era which had nothing like it).

Also invisible to the player are the strength levels of the various creatures, the amount of damage that the weapons inflict, the time left for the burning torch, and a few other things. There have been suggestions to make these types of things visible in the port, and this is certainly possible to do as a future enhancement, but the port will initially be just like the original and will not show any internals.

For myself, the mystery was overwhelming, and I purchased a copy of the source code from the author primarily so I could finally learn everything about the game that I had always wanted to know. This page attempts to give answers to the most commonly asked questions. If you have specific questions that are not addressed here, please post them in the forum, and as time permits, I will add the answers here (if I can).

Heartbeat

The most logical place to start is with the heartbeat, since that is what the player can actually see. The player has two attributes which together determine the heart rate: power (PPOW) and damage (PDAM). When the game begins, the player's power attribute is set to 160, and the damage is zero. The power never decreases, but it can increase by killing monsters and by drinking a thews flask. The damage level is increased by doing various different things: walking, attacking, getting hit, drinking an abye flask. When the damage level exceeds the power level, the player dies, and the game is over, and that most annoying of all messages from the wizard flashes on the screen: YET ANOTHER DOES NOT RETURN.

The heart rate itself is calculated using the JIFFY-delay formula shown below, which results in the number of JIFFYs that have to pass before the heart beats again. A JIFFY (as the source code calls it) is 1/60th of a second. On the original CoCo, the CPU received a hardware interrupt 60 times per second, so it was natural for the game designers to use this to measure time. To find out beats per minute, divide 3600 by the result of the JIFFY-delay formula. In the code, whenever the JIFFY delay hits 3 (1,200 BPM), the player faints.

JIFFY-Delay Formula, as shown in source:

$$HEARTR = \frac{PPOW * 64}{PPOW + (PDAM * 2)} - 19$$

In the DoD programmers' implementation of the division algorithm used by this formula, it adds a one to the integer quotient, i.e., 1/5 = 1, 5/5 = 2, 10/5 = 3. Therefore, the formula should really subtract 18 to achieve the same number as in the original implementation.

Each time the heart beats, there is also a damage recovery routine that lowers your damage level using this formula: PDAM = PDAM - (PDAM / 64), resulting in a slowing of the heart rate. Because of this, whenever you are standing still (and not getting hit by a creature), your heart rate will slow down, eventually reaching approximately 78 BPM. This formula will never result in a complete recovery from damage. The only way to get your damage back to zero is by drinking the hale flask.

Increasing Power

Power is increased by two means: the thews flask, and killing creatures. The thews flask simply increases your power by 1000 points, which, if your heart is racing, will result in a slowing of the heart rate somewhat. Also, each time you kill a creature, your power is increased by 1/8 of the power rating of the particular creature you killed. So essentially you absorb the power from your defeated foes.

The creatures' power ratings (and initial number of creatures per level) are as follows:

Creature	Power	Level 1	Level 2	Level 3	Level 4	Level 5
Spider	32	9	2	0	0	2
Viper	56	9	4	0	0	2
Stone Giant 1	200	4	0	0	0	2
Blob	304	2	6	4	0	2
Knight 1	504	0	6	0	0	2
Stone Giant 2	704	0	6	6	0	2
Scorpion	400	0	0	8	8	2
Knight 2	800	0	0	4	6	2
Wraith	800	0	0	0	6	2
Galdrog	1000	0	0	0	4	8
Wizard's Image	1000	0	0	1	0	0
Wizard	8000	0	0	0	0	1

This means that at the end of level one, if you have killed all the creatures, your power will be equal to 160 + (32 / 8 * 9) + (56 / 8 * 9) + (200 / 8 * 4) + (304 / 8 * 2) = 435. And, at the end of the game, if you have killed all the creatures except the wizard (not counting extra ones after climbing up a level), you will have a power rating of 7,260, and after killing the wizard, 8,260. If you climb up a level, it is possible to kill more creatures (see below). However, the player's power can never go higher than 32,767, which would take a very, very long time to reach (you would have to kill the equivalent of an extra 197 galdrogs). There are also the three thews flasks (one each on levels 3, 4, and 5 -- see below), which together will give you an extra 3,000 power points (which can shave 24 extra galdrogs off the required number to hit the 32K power cap).

Creature Speed

Every creature in the dungeon has two speed ratings (implemented as delay timers), one for movement and one for attack. The attack rating is used whenever the creature is in the same room as the player, and the movement rating is used in all other cases. The table below shows the various creature speeds (i.e., delays) in tenths of seconds:

Creature	Movement	Attack
Spider	23	11
Viper	15	7
Stone Giant 1	29	23
Blob	31	31
Knight 1	13	7
Stone Giant 2	17	13
Scorpion	5	4
Knight 2	13	7
Wraith	3	3
Galdrog	4	3
Wizard's Image	13	7
Wizard	13	7

These values are important for understanding the relative speed of each creature compared to the other creatures. Also, for the port, this table shows why the creatures seem to move so fast. On the original hardware, the wraiths probably did not move 3.3 times per second, but that is clearly the intent of the code. Since our modern platforms are so much faster (and therefore the scheduler will work more accurately), each creature's movement will probably more accurately reflect the "authorial" intent of the code than it did on the original hardware.

Combat

Combat in Daggorath is performed in a set of two routines: ATTACK, and DAMAGE. The same calculations are used for both a player attacking a creature and a creature attacking a palyer. Each creature type has predefined values for magical offense and defense and physical offense and defense. For the player, the magical and physical offensive values are determined by the weapon being used to attack. Each object type has both magical and physical attack values. The player's defensive values are determined by the particular type of shield being held.

The ATTACK routine determines if an attempted attack strikes its target. If there is a hit, then the DAMAGE routine calculates and inflicts damage. The ATTACK routine uses some funny logic, and a bit of randomness. Based on a comparison of the power of the attacker versus the remaining life force of the defender (power minus damage), the attacker has between a 21% and 97% chance of striking the defender. If a strike occurs, then for players attacking creatures, there is an additional caveat that if the player has either no torch lit or a dead torch, then, of those times when ATTACK returns a hit, only 25% will actually hit the creature. Creatures, however, can hit in the dark without any penalty.

The chart below shows the probability curve between the Strength of the attacker (percentage of attacker's power versus defender's remaining life force) and the Probability of the ATTACK routine returning a successful hit. As you can see, there is always at least a 1 in 5 chance of hitting the target, no matter how outmatched you are. Likewise, there is always at least a 3% chance of missing the mark, regardless of how much stronger you are than the one you're attacking.

Strength	Probability
<= 25%	21%
33%	50%
66%	75%
100%	85%
200%	92%
> 400%	97%

When there is a hit, the DAMAGE routine is called. It uses the attacker's power, the defender's remaining life force, and the four values of magical and physical offense and defense. The magical and physical offensive and defensive values are stored as radix-7 binary real numbers (b.bbbbbbb). They act as multipliers, and can be usefully thought of as percentages. Their range is 0% to 199% (or as rational numbers: 0/128 to 255/128). A high offensive multiplier is good for the attacker, and a low defensive multipler is good for the defender. So the creatures with high offense and low defense are very powerful, and the ones with low offense and high defense are very weak. The tables below show the various numbers for the creatures, and for the objects (only shields have defensive values).

Creature	Magical Offense	Magical Defense	Physical Offense	Physical Defense
Spider	0 (0%)	255 (199%)	128 (100%)	255 (199%)
Viper	0 (0%)	255 (199%)	80 (63%)	128 (100%)
Stone Giant 1	0 (0%)	255 (199%)	52 (41%)	192 (150%)
Blob	0 (0%)	255 (199%)	96 (75%)	167 (107%)
Knight 1	0 (0%)	128 (100%)	96 (75%)	60 (47%)
Stone Giant 2	0 (0%)	128 (100%)	128 (100%)	48 (38%)
Scorpion	255 (199%)	128 (100%)	255 (199%)	128 (100%)
Knight 2	0 (0%)	64 (50%)	255 (199%)	8 (6%)
Wraith	192 (150%)	16 (13%)	192 (150%)	8 (6%)
Galdrog	255 (199%)	5 (4%)	255 (199%)	3 (2%)
Wizard's Image	255 (199%)	6 (5%)	255 (199%)	0 (0%)
Wizard	255 (199%)	6 (5%)	255 (199%)	0 (0%)

Object	Magical Offense	Magical Defense	Physical Offense	Physical Defense
Wooden Sword	0 (0%)	-	16 (13%)	-
Iron Sword	0 (0%)	-	40 (31%)	-
Elvish Sword	64 (50%)	-	64 (50%)	-
Leather Shield	0 (0%)	108 (84%)	10 (8%)	128 (100%)
Bronze Shield	0 (0%)	96 (75%)	26 (20%)	128 (100%)
Mithril Shield	13 (10%)	64 (50%)	26 (20%)	64 (50%)
Any Torch	0 (0%)	-	5 (4%)	-
Any Flask	0 (0%)	-	5 (4%)	-
Any Scroll	0 (0%)	-	5 (4%)	-
Unincanted Ring	0 (0%)	-	5 (4%)	-
Incanted Ring	255 (199%)	-	255 (199%)	-
Gold Ring	0 (0%)	-	5 (4%)	-
Empty Hand	0 (0%)	-	5 (4%)	-

Each creature type has its combat multipliers hard-coded. For the players, however, they are calculated each time a combat situation occurs. When a player attacks a creature, the object in the attacking hand determines the magical and physical offense values. When a creature attacks, the player's defensive values get calculated by first assigning the default, no-protection value of 128 (100%) [equivalent to multiplying by 1] for both magical and physical. Then each hand is checked, and if a shield is found, then it's values are used instead. If there are two shields, then the better of the two is used. The formula actually concatenates the two binary numbers together, with the magical value in the most significant byte, and then chooses the lowest (i.e., best) option. This has the effect of giving the magical value precedence over the physical value in making the determination.

The DAMAGE routine takes the attacker's power and multiplies it by the attacker's magical offense, and that product is multiplied by the defender's magical defense, resulting in the magical damage. Then the same is done with the physical offense and defense values. Then the magical and physical damage values are added together, and the sum is added to the defender's damage value. Then it checks if the defender's damage has become greater than the defender's power, and if it has, then the defender dies. So the formula looks like:

DDAM = DDAM + (APOW * AMO * DMD) + (APOW * APO * DPD).

Understanding the way Daggorath handles combat makes it clear why certain things in the game happen the way they do. For example, the wizards cannot suffer physical damage because both have physical defense values of zero, which means that any physical damage you try to inflict gets multiplied by zero. The creatures with zero magical offense, don't inflict any magical damage. Galdrogs and wizards are so hard to kill because no matter how powerful you are, they only suffer a very small percentage of any damage you try to inflict. The only weapons that inflict magical damage on creatures are incanted rings, the elvish sword, and the mithril shield.

If you examine the above closely, there is a curious thing to note about the shields. You may be thinking that I got the numbers backward, but those are the actual numbers from the code (both the binary ROM and the source code). The strange thing is that the leather and bronze shields do not protect against physical damage at all, but they do protect against magical damage. This explains why many people are convinced that the shields don't help any, because in these cases, they actually don't. I believe that this is an error in the game. It would make so much more sense to have the leather and bronze shields protect against physical damage but not magical. This would be in line with the way the swords work, with the wooden and iron having no magical offense but only physical. If anyone has a different interpretation, I would love to hear it. I have made the port act exactly the same way as the original in this matter, even though I'm sure it must be an error. However, there could be an optional shield fix configuration added.

Suffering Damage

Damage is inflicted each time you get hit, as explained above, and also each time you attack and each time you take a step -- and of course by the abye flask. (1) The abye flask results in your damage increasing by 80% of you power: PDAM = PDAM + (PPOW * 0.80). As should be clear, this can very easily make your damage exceed your power, resulting in instant death. It could also very easily cause you to faint. Therefore, you should never drink this unless your heartbeat is very slow. (2) Any time you attempt to attack, regardless of whether you hit anything or not, you suffer damage equal to the your power multipled by 1/8 of the sum of the magical and physical offensive values of the weapon in the hand you're attacking with (or the empty hand values). (3) Each time you MOVE (but not when you TURN) you suffer damage equal to 1/8 of the weight of the objects you carry plus three. The object weights are shown in the table below. Your initial object weight is 35 (sword and torch), and it gets updated accordingly each time you GET or DROP an object. When you get transported to the fourth level, your object weight value is set to 200 (regardless of what you're carrying) as a penalty.

Object	Weight
Flask	5
Ring	1
Scroll	10
Shield	25
Sword	25
Torch	10

Object Distribution and Revelation

Objects are distributed to the different levels at the beginning of the game, and they are assigned to the creatures on their respective levels each time that level is entered by the player. Only one level is stored in memory at a time so the current level has to be rebuilt each time the player changes levels. The objects are distibuted at the beginning of the game according to the table below. The objects never regenerate. Their quantities remain constant throughout the game. They can of course be carried to different levels by the player. And, as we all know, the player is given a revealed wooden sword and pine torch at the beginning of the game.

At the beginning of the game, all the objects (except the ones the player is carrying) are marked as creature-owned, and their dungeon-level is marked as indicated in the table below. Each time a level is built (or rebuilt), all the creatures on that level are assigned all the creature-owned objects on the level. And the way it works is that the most powerful creatures end up getting the most powerful (or important) objects. The table below lists the objects in their importance ranking. This can cause weird things to happen if you climb up a level (see below).

The table below also lists the required amount of power the player must have to reveal an item. At the end of the first level your power can be at most 435, but the revelation requirement for the lunar torch is 625, which explains why you can't yet reveal it. Likewise, at the end of the second level (provided you have not climbed up and killed extra creatures), your power can be at most 1,605, but the solor torch's revelation requirement is 1,750, hence the inability to reveal it.

Also note that there is a seer scroll on the third level, which has a higher ranking than any other objects that can occur on that level, and so the wizard's image (the most powerful creature on the level) always gets the seer scroll (the most important object on the level) distributed to him. This is the scroll that you see drop on the floor when you kill him, but which you can never pick up because you are immediately transported to the fourth level. It's a good thing too. If he didn't have a high-level object like that then he would instead have the very important thews flask, but you would never be able to get it (as you can't get his scroll).

Object	Revelation Requirement	Level 1	Level 2	Level 3	Level 4	Level 5
Supreme Ring	6,375	0	0	0	0	1
Joule Ring	4,250	0	0	0	1	0
Elvish Sword	3,750	0	0	0	1	0
Mithril Shield	3,500	0	0	0	1	1
Seer Scroll	3,250	0	0	1	1	1

Thews Flask	1,750	0	0	1	1	1
Rime Ring	1,300	0	1	0	0	0
Vision Scroll	1,250	0	1	1	1	0
Abye Flask	1,200	0	2	2	1	1
Hale Flask	1,000	0	1	1	1	1
Solar Torch	1,750	0	1	1	1	1
Bronze Shield	625	0	2	2	1	1
Vulcan Ring	325	1	0	0	0	0
Iron Sword	325	1	1	1	1	0
Lunar Torch	625	2	2	2	1	1
Pine Torch	125	2	2	2	1	1
Leather Shield	125	1	1	1	0	0
Wooden Sword	125	1	1	1	1	0

Torch Lighting

Torches in Daggorath shed both physical and magical light. The creatures with magical offensive values greater than zero (scorpions, wraiths, galdrogs, and the two wizards) can only be illuminated by a torch that throws magical light. The same is true for the magic (or secret) doors. The table below shows for each torch the number of minutes it will burn and the amount of physical and magical light it will cast. The way this is implemented is that once per minute, a routine gets called that decrements the active torch's timer. The luminescence numbers can never be higher than the timer, so they decrease also once the timer has gotten down to their level. When the timer hits five minutes, the torch chages to a dead torch, but it continues to cast an ever decreasing amount of light until it hits zero, when it will cast no light at all. If you pull an active (lit) torch from the backpack into your hand, you essentially extinguish it, but if you use (light) it again, it will continue on using the same numbers it had when you extinguished it.

Torch Type	Minutes	Physical Illumination	Magical Illumination
Pine Torch	15	7	0
Lunar Torch	30	10	4
Solar Torch	60	13	11

So, when you light a lunar torch, it shines at its full brightness for 20 minutes. When the timer drops to 9, the physical lighting will also drop to 9, and will continue to drop in step with the timer. When it hits 5 minutes, it turns into a dead torch. When it hits 3 minutes, the magical lighting value drops to 3 (as does the physical). At zero, all the values are zero, and the torch is totally dark.

After you kill the wizard, all objects disappear from the dungeon except the supreme ring, including whatever torch you may have had lit. When this happens, ambient lighting values are set for the dungeon, equivalent to a torch with a physical lighting value of 7 and a magical value of 19.

Generic Objects

Torches, swords, and shields, which have not been revealed have generic numbers, meaning they have the values associated with the lowest member of their class. An unrevealed iron or elvish sword will act just like a wooden sword, and the same with unrevealed bronze and mithril shields, which act just like leather shields. For torches, this has a special consequence. Unrevealed torches will burn just like pine torches, and all their numbers will get reset when they get revealed. So you can burn an unrevealed torch for 9 minutes (and it will shed the same light that a pine torch sheds), and then you can reveal it, and the numbers will get set to their proper values according to the table above. But if you let the unrevealed torch's timer get down to five minutes, it will turn into a dead torch, and then you will not be able to reveal it, because its object type will have changed (i.e., it's no longer an unrevealed lunar torch, it's now a revealed dead torch). This generic behavior essentially gives you 9 extra minutes of pine-torch lighting for every torch (including unrevealed pine torches) that you find in the dungeon.

Flasks

To summarize what has been mentioned above. The three flasks do the following:

- **Hale Flask:** Resets the player's damage value to zero.
- **Abye Flask:** Increases the player's damage value by 80% of his or her power value.
- **Thews Flask:** Increases the player's power value by 1000 points.

Dungeon Maze Generation

The algorithm for creating the mazes is pretty interesting. You start out with a 32 x 32 square block, with each square having four walls. Then a random square is chosen as the place to start "digging tunnels." Then a random direction is chosen and a random number of squares to go in that direction, and then the tunnelling process carves out that many squares by removing all their walls. This is repeated until the total number of squares carved out equals 500, which is just under half of the total possible squares (1,024). Once all the tunnels are carved, then 70 regular doors and 45 magic doors are randomly placed throughout the maze.

The dungeon is stored in 1,024 consecutive bytes of memory, with each room being defined by a single 8-bit integer, with 2 bits for each of the four directions. 00 means an open corridor, 01 is a regular door, 10 is a magic door, and 11 is a solid wall. Therefore, all the spaces not in the 500 carved out spaces are all walls: 11111111. The two least significant bits, represent the north direction, the next two are the east, the next are the south, and the two most significant bits are the west. So a room with an open corridor on the north, a wall on the east, a regular door on the south, and a magic door on the west is stored as: 10011100, which is decimal 156.

The holes and ladders are stored in a separate table called the Vertical Features Table, and were hard-coded, as explained in the following section, after the random number generator seeds were decided upon.

Random Numbers

The Daggorath programmers wrote (or borrowed) their own pseudo-random number generator routine. It is a pretty standard feedback-shift algorithm which uses three 8-bit seeds, shifts them, counts the number of 1's, and then adds that count to the original, then returns the first seed in its now modified state. At the end of the process that builds each level, the RNG is spun a number of times equal to the SECOND counter in the CLOCK routine. This has the effect of causing the first level of creatures, to always start in the same place when you first start the game, because the SECOND counter has not yet begun counting. However, on the subsequent levels, the creatures' positions are truly random. The RNG is also used when a creature has to move, but cannot "see" the player, so has to choose a direction, in a planned, but somewhat random fashion. And it is used to decide if an attempted attack strikes its target or not.

The dungeon mazes themselves are built using the RNG, however since the seeds are set to predetermined values at the beginning of each maze build, the mazes always come out the same. The DoD programmers must have tried out several seeds and eventually chose the ones that made it into the final version. How they made this decision is a mystery I would love to find the answer to. The holes and ladders, however, are hard-coded, and must have been placed only after the seeds had been decided. They would have had to find spots where the corridors on two adjoining levels matched up. Why they did it this way is hard to say, but it was probably to save space. However, it seems clear that originally the game was intended to have random mazes that were not the same every time you played the game, but this feature was cut out of the final version. [See posts by Jim Thomas (one of the original Daggorath programmers) in the Fourm, for explanations about the random mazes.]

Why Climbing Up Makes Weird Things Happen

The game initially sets up how many of each creature type are on which levels (see table above). Each time the player enters a new level, this table is used to generate the appropriate number of each type of creature. Whenever a creature is killed, its corresponding counter for that level is decremented. However, once every five minutes a routine runs which randomly increments one of the counters for the current level (except for spiders, vipers, and the wizards). This has no immediate effect, but if this same level gets entered again (by climbing up), then its corresponding creatures get recreated according to the table (which now has different numbers in it than it did the first time it was built).

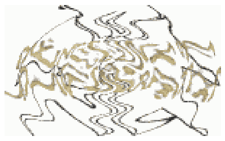
So the typical way this happens is that the game starts on level one, and the player goes through and kills all the creatures, resulting in all the counters being decremented to zero. However, during this process, every five minutes, one of the counters gets randomly chosen to be incremented (so that at the end they aren't really all zero). Then the player climbs down to level two, and fights for a while, and then decides to go back up to level one. Now when level one gets rebuilt, it uses the numbers in the table, which are completely different than they were when the game began, and might have had the counters for wraiths or galdrogs incremented, or knights, or scorpions, stone giants, or blobs.

Also, when any level gets built, any objects that are marked as creature-owned (i.e., not on the floor or in the player's backpack), are distributed to the creatures on the level, with the strongest creatures getting the best objects. Because of this, strange things happen if you don't kill all the object bearing creatures the first time through a level. For example, if you don't kill the blob with the ring on level 1, and then later climb back up to level one, and if the galdrog counter has gotten randomly incremented, then when level one gets rebuilt the second time, the galdrog (being the strongest) will have the ring, and the blob (which is still there since you didn't kill it) may have a torch or shield or nothing.

Questions, Comments, and Feedback

I'm sure I've left out some important and / or interesting stuff. If you have questions, please post them to the forum Internals tread, and I will do my best to add the answers here. Also, if you notice any mistakes, or have any comments or feedback (good or bad), please post those as well. Thanks!

Home



May 4, 2006
Richard Hunerlach (Personal Page)

(Animated GIFs created by Bruce)