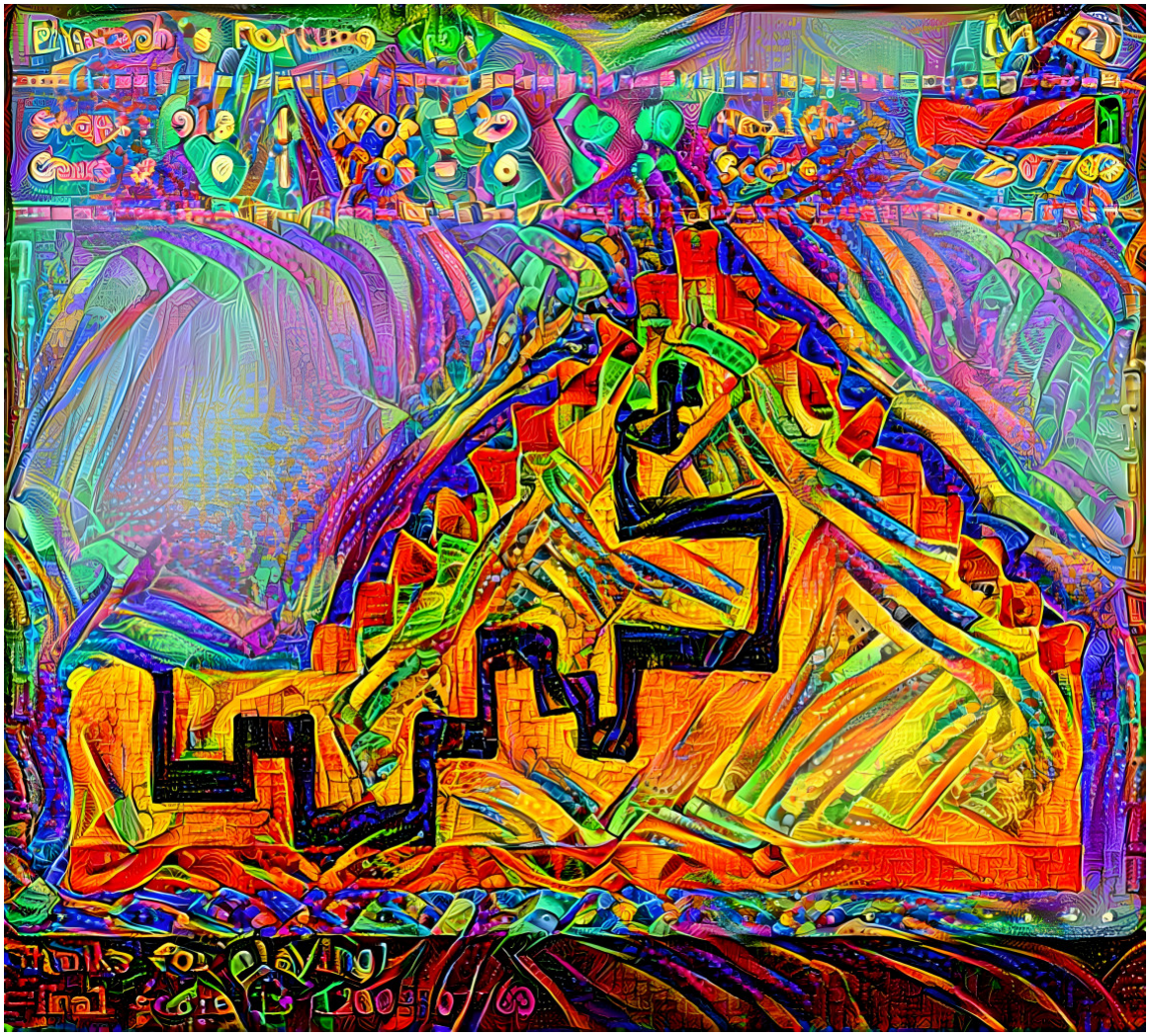
Pharaoh's Fortune

Player Guide — ver. 1.444



Contents

1 Getting Started

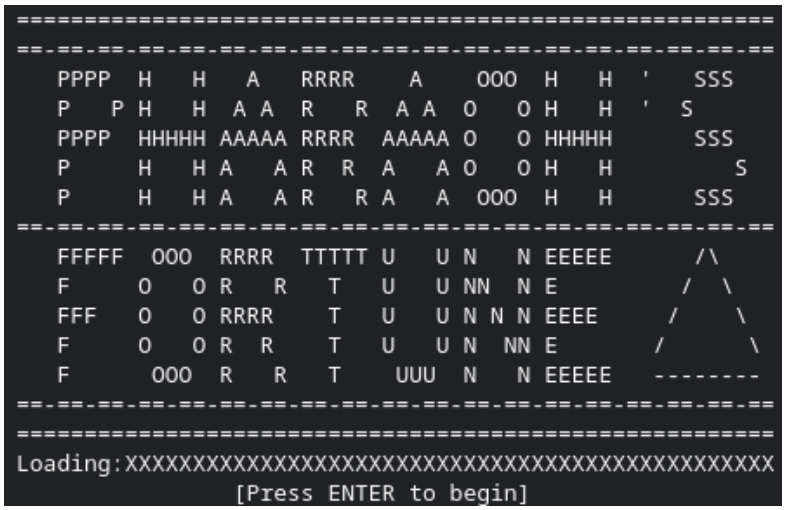
$\mathcal{W}$elcome to Egypt, explorer! To get playing, first open up a terminal. Hopefully, you're greeted with something like

```
blip@blam:~$
```

If not, you have much bigger issues! Assuming all is well in Denmark, type

```
blip@blam:~$ bash phar_fo.sh
```

And that's it! If everything is dandy[1], you'll see



Now, just press [ENTER] to start the game! Alternatively, type in one of the following cheat codes to adjust your experience...

Input	Effect
`khu-***`	Larger pyramid
`****Ra`	Enable All-Seeing Eye
`ALL_******`	+666 gems
`j*****!`	Double maximum health
`deathwish`	9 × cursed
`****nell`	Full inventory + amulet
`**`	Full inventory, 999 gems, 99 spells, amulet × 9 + All-Seeing Eye

Table 1: Summary of cheat codes. [*Note that asterisks are not part of the input.*]

...and *then* press [ENTER]. After that, it's up to you! See §2 to learn what you might dig up, and §3 to understand how to use it. The next section (§4) discusses what's required to unlock the Pharaoh's sarcophagus, and how to maximize your final bonus. Finally, §5 talks some strategy. *Thanks for playing!*

[1]There are two reasons to be poking around down here. One, you just can't help but read irrelevant footnotes, or two, everything is definitely *not* dandy. If it's the latter, see §5.

2 Navigating the Ancient Pyramid

$\mathcal{A}$ successful game of *Pharaoh's Fortune* will invariably depend on one's interaction with the overworld. To that end, the general layout, HUD, and controls are presented in Figures 1 & 2. The remainder of this section outlines the various treasures to be extracted from the sand, so... get digging! (Eventually, we'll concern ourselves with finding the Great Pharaoh, but for now it behooves us to simply get comfortable!)

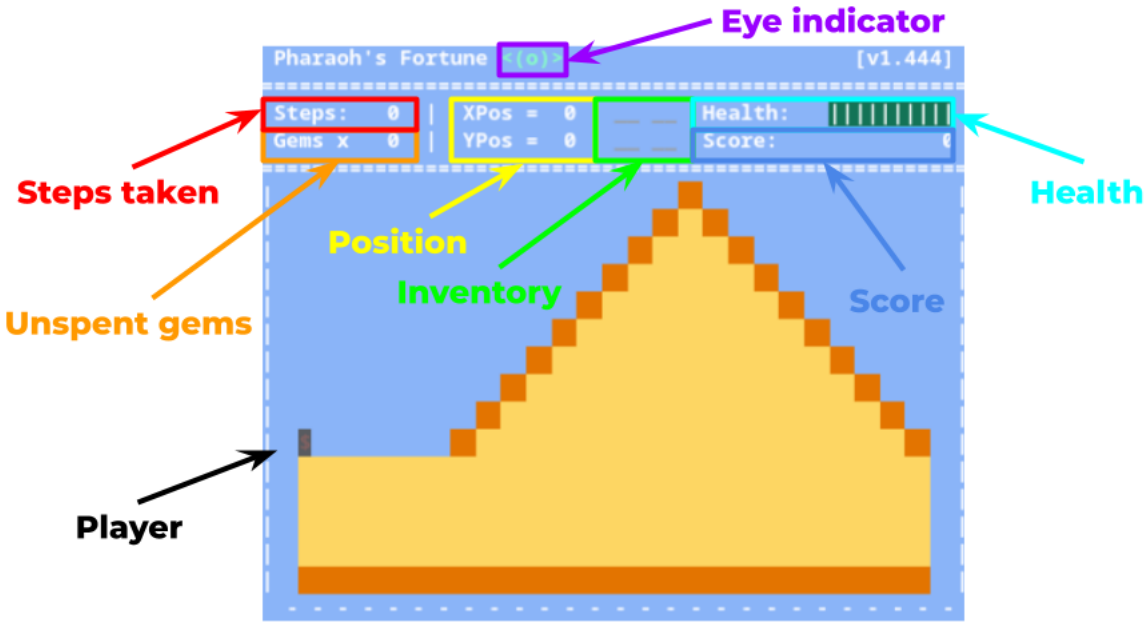


Figure 1: Layout of pyramid, player, and HUD.

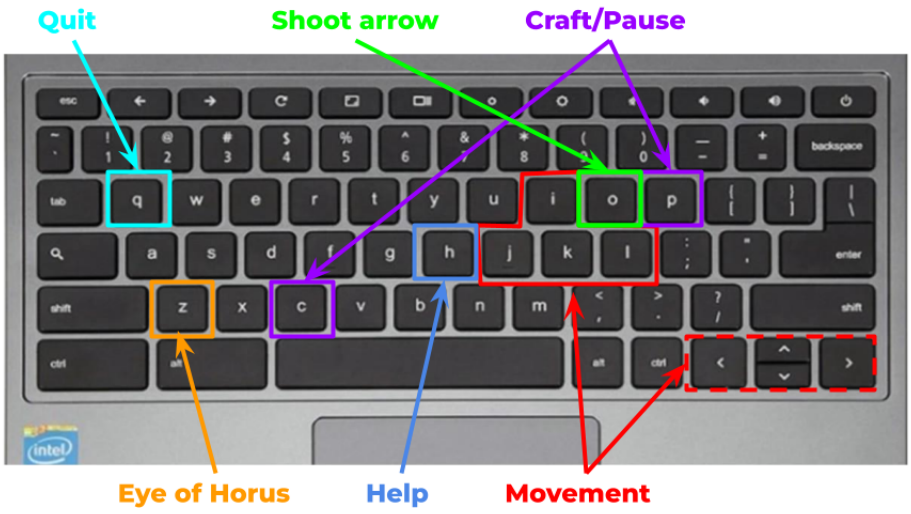


Figure 2: Overview of standard keyboard configuration. Note that the arrow keys are supported, but depreciated.

2.1 Basic controls

Input	Effect	Item required
I/J/K/L	Move up/left/down/right	none
C/P	Go to crafting table	none
H	Go to help menu	none
Q	Quit game	none
Z	Toggle All-Seeing Eye on/off	spell
O	Shoot arrow	bow+gunpow

Table 2: Summary of inputs for main map.

2.2 Gems

Gems are the *de facto* currency in the land of the Pharaoh. Each one you discover yields +1 gem (+3 if wearing a ring), and 100 points. Use this booty to bankroll your crafting sprees!

2.3 Mummies

If you were an embalmed individual trapped within a cursèd compartment, how would you choose to greet your first visitor in 4,000 years? In the land of the Pharaoh, there is only one answer — *attack first and ask questions later*. True to form, uncovering a mummy will result in −1 HP (unless you have crafted a dagger), and earn you 500 points. After the attack, the mummy poses a random question relating to ancient Egypt[2].

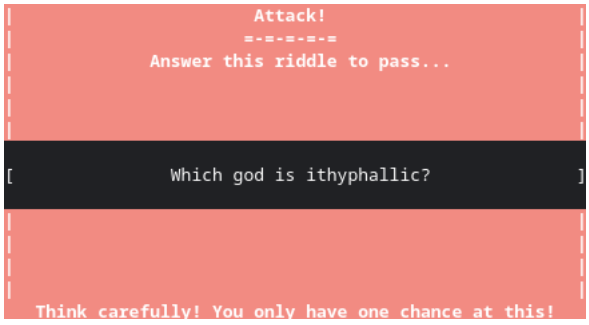


Figure 3: Depiction of typical question as prompted by mummy encounter.

To attempt an answer, just type in your guess, as

```
myGuessHere
```

and smack [ENTER][3]. If you're correct, you earn 3 gems and 1,500 points — if you meet enough mummies to see all their questions (currently 20), you receive an additional 100,000 points!

[2]Well, for the mummy, there's nothing "ancient" about it. They'll be asking about their impression of *modern* Egypt.
[3]There's no need to worry about case-sensitivity here! All input is converted to lower-case before parsing.

2.4 Curses

Spoken into the stone by the gods themselves, the many curses within the pyramid walls seek a singular purpose — eliminate all intruders. Thus, the gods will recoup your debt of blood, in proportion to the number of curses the player is juggling[4]. Luckily, due to the fickle nature of these curses, items under the sand will become visible to the player, with the radius of detection also proportional to the number of stacked curses (see Figure 4).

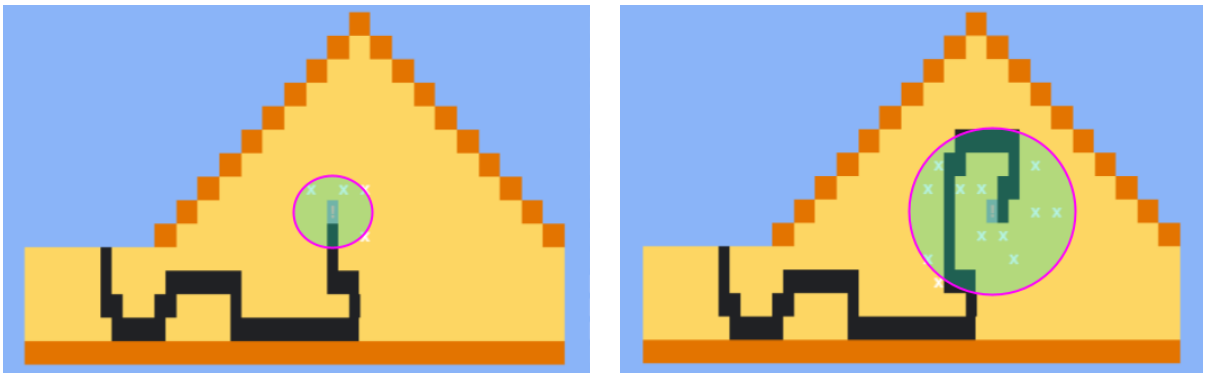


Figure 4: Detail of overworld, with highlighted area of detection, for [*left*] 1 curse, and [*right*] 3 curses.

These effects are summarized by the following table[5]:

# of Curses	Safe steps before −1 HP	Visible radius
1	3	1
2	2	2
3	1	3
$n \geq 4$	0	n

Thus, when carrying 4 or more curses, the player will lose 1 HP on *each step taken*. To compensate, the player's score increases with each cursed step, as

$$\Delta(\text{score}) = 13 \times (\#\text{ of curses})(\#\text{ of consecutive cursed steps}),$$

so extended jaunts with the darkness will provide a veritable windfall.

2.4.1 The Eye of Ra

If the player can survive long enough to collect 6 curses, a very special thing happens. Acknowledging your dark allegiance (and lack of fear!), the gods lend their blessing through the Eye of Ra. Though you will continue to lose HP on each step, the dark Eye **allows spells to be crafted for 5 gems**. This mechanism defines the so-called *path of Ra*, where a player retains their curses to obtain spells for a 90% discount [see §3.9]. As spells also provide a massive end-game bonus, this path is the essential risk/reward in the land of the Pharaoh. If you choose to walk the cursèd way, keep in mind:

$\Longrightarrow$ Ankhs are invaluable! It is unlikely you will survive without crafting many!
$\Longrightarrow$ Rings and jars provide the gems and health, respectively, that are necessary to keep moving!
$\Longrightarrow$ Good showings in booby traps are huge! Do better, live longer, get a bigger score!

[4] Accumulating 10 curses will *immediately* kill the player. Do not pass Giza, do not collect 200 ankhs.

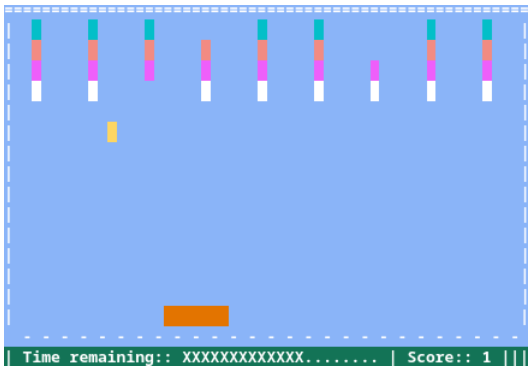
[5] Here, "radius" is used for visible distance in Y. Strictly speaking, the visible distance in X is 2×radius $= 2R$, due to the cursor being approximately twice as long as it is wide. The explicit formula is $(\Delta X/2)^2 + (\Delta Y)^2 < R(R+1)$.

2.5 Booby Traps

Pharaoh's primary method of intruder murder, booby traps challenge the player to think fast, react quickly, and strategize in real-time. Despite the danger, these perilous performances can prove exceptionally lucrative, and generally serve to facilitate the opening of Pharaoh's sarcophagus. *Take that, expectations!*

2.5.1 Mudbrick Break-out

Inspired by an ancient extraterrestrial simulation, the player finds themselves buried under a mountain of mudbricks. The inexplicable solution to this dilemma is breaking things with a bouncing stone. The player receives a short amount of time to destroy as many bricks as possible. **Dropping your stone before the time limit will remove one HP, as will breaking a white brick**. *Each mudbrick you break will contain a varying amount of gems.*



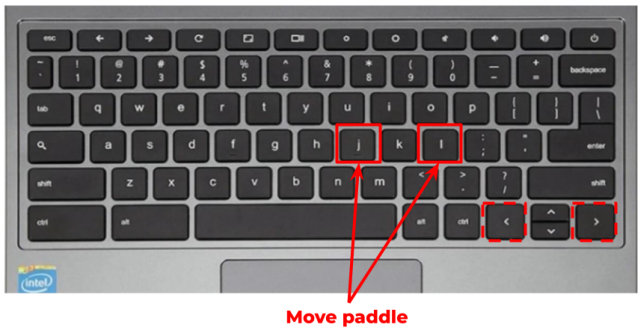


Figure 5: [*left*] Screenshot of gameplay, [*right*] explanation of brickbreaker controls.

2.5.2 Snake Pit Survival

Did you wake up in a roiling river of serpents? Have no fear! Just pretend to be a snake, and eat snake eggs before they start hatching. (Snake babies look like a sour-colored snake egg, and are just as nutritious!) **You will lose one HP if you bite yourself, or fail to stop the current egg from hatching.** *Alternatively, everything you eat will be somehow be processed into a gemstone!*

Figure 6: [*left*] Screenshot of gameplay, [*right*] explanation of snake controls.

2.5.3 Salt Acid Escape

A classic way to die: You fall through the floor, dusting yourself off in a dark room filled with earthenware vats. The catch? Some of these clay canisters contain pressurized salt acid. Luckily for you, there is a placard under each lid detailing how many nearby vats contain acid. (A fool-proof system if there ever was!) By moving around the room, you can flag vats with either **?** for unknown, or **F** for "fuck that!" *You will earn a single gem for each correctly identified vat of acid.* This booby trap, while tedious, is surely profitable. **A single HP is lost if you open the wrong vat, and for each incorrectly identified amphora.**

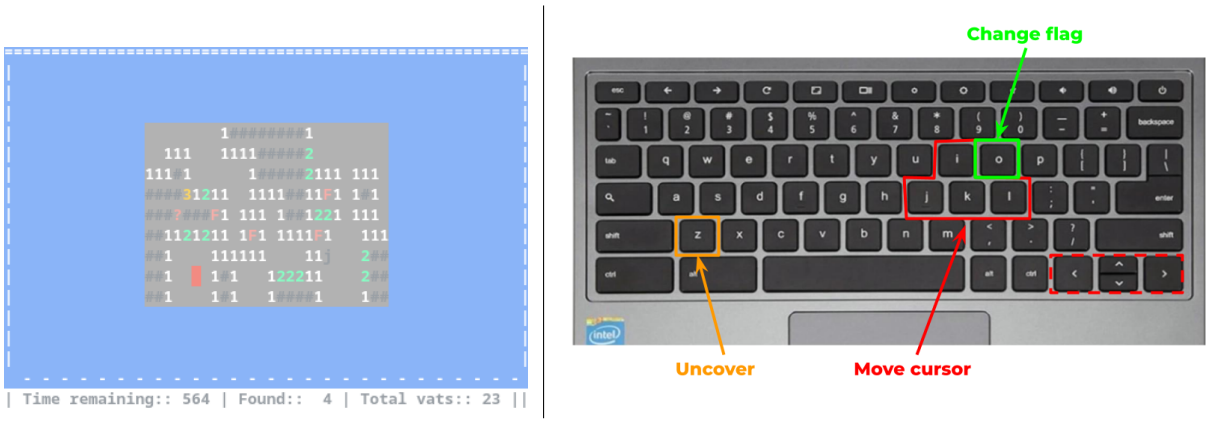


Figure 7: [*left*] Screenshot of gameplay, [*right*] explanation of minesweeper controls.

2.5.4 Ancient Alien Attack

Thanks to your rediscovery of extraterrestrial technology, visitors from Nebulon IX are here to take back what's rightfully theirs. Do Earth a favor and shoot these complete strangers in the face with your laser bow! Will Giorgio Tsoukalos have a problem with you destroying an advanced interstellar society, just so you can keep your shiny shooty-stick? Probably. Will the inescapable decay of human civilization rest squarely on your sandy shoulders? Sure. But that's the price of having cool shit, my dude! During battle, you may fire up to 3 shots at a time, all the while attempting to evade the visitors' purple lasers. **As the invaders' tactics and technology far exceed your own, being shot will end the attack, and remove one HP from the player.** *Destroying any alien craft yields gems, with higher ships being worth more.*

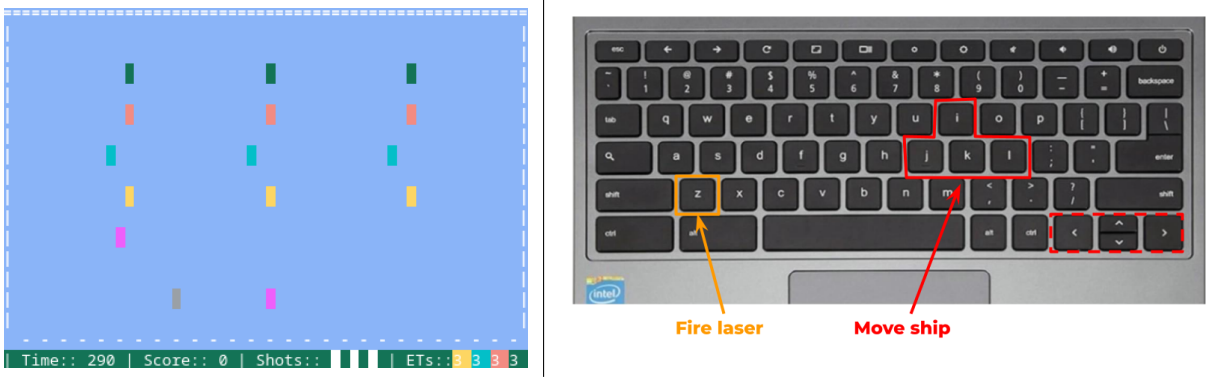


Figure 8: [*left*] Screenshot of gameplay, [*right*] explanation of "Galaga" controls.

3 Crafting your Livelihood

Once you accrue a cache of gems, you'll eventually want (or probably *need*!) to increase your chances of survival by crafting a few useful items. To access a workbench, press either [C] ("craft") or [P] ("pause"), as both will bring you to the menu shown in Figure 9.

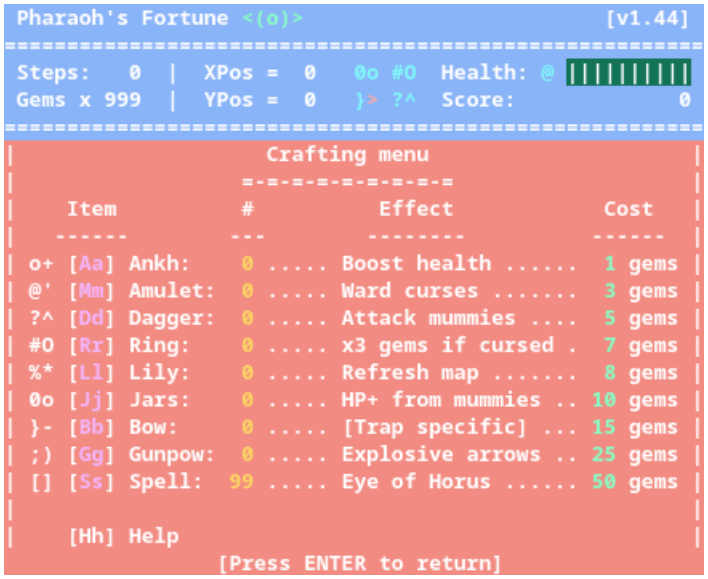


Figure 9: Reproduction of crafting menu.

To craft a number of items, use a command of the form

```
[item] [number]
```

At minimum, the `[item]` field must contain the first letter of the desired item (with the exception of a**M**ulet!), but the full word is also acceptable. If the `[number]` field is blank, `number=1` is assumed. Thus,

```
m3
M 3
mulet 3
```

all result in 3 freshly-crafted amulets[6]. Using items is just as simple: the command looks like

```
[item] -use [number]
[item] -u [number]
```

Hence, each of the following inputs uses 5 of the player's ankhs:

```
a -u 5
A 5 -u
ankh -use 5
```

Conveniently, the `-u` and `-use` flags **automatically craft the item**, if needed.

[6] Assuming you have enough gems to cover the balance, of course!

3.1 Ankh (Aa)

[o+] **(1 gem)** Your **primary means of recovering health**, sintered quartz (faience) ankhs provide a cheap way to stay in the game. *If you choose to walk the path of Ra, crafting and using multiple ankhs will be a common occurrence.*

3.2 Amulet (Mm)

[@'] **(3 gems)** Protective necklace made with yellow-green desert glass. If uncursed, player receives (additional) protection equal to the number of amulets crafted (**max = 9**). Otherwise, an amulet immediately **wards any level of curse**.

3.3 Dagger (Dd)

[?^] **(5 gems)** Razor-sharp and forged from meteoric iron. In deference to its sky-borne blade, **mummies will not attack the player when encountered**. *Can be quite cost-effective if used early!*

3.4 Ring (Rr)

[#O] **(7 gems)** Made from lapis lazuli and pure gold, this ring will compel the gods to bless your accursed endeavors. When cursed, wearing a ring will **triple the value of all gems found** within the pyramid. To be clear, this *will not affect gems gained from mummies or booby traps.* Does **not** stack.

3.5 Lily (Ll)

[%*] **(8 gems)** A large, pure-white water-lily, noted for its beauty — and intense narcotic properties. When ingested, the former reality is washed away, and **the player returns to a fresh map**, devoid of mar or blemish. *It is generally unlikely that a player could open Pharaoh's sarcophagus without use of a lily.*

3.6 Jars (Jj)

[Oo] **(10 gems)** Alabaster canopic jars containing 4000 year old organs. When the player possesses these artifacts, **mummies will provide +2 HP in exchange for a correct answer to their query**. Does **not** stack.

3.7 Bow (Bb)

[}-] **(15 gems)** A composite bow crafted with acacia wood, hoof-glue, a mysterious bone, and human sinew. Originally designed by extraterrestrials in 2222 BC. Its beauty is such that visitors appear in hopes of reverse-engineering their former glory. This **allows the player to access the "Ancient Alien Attack" booby trap**. *Confers various benefits in all other booby traps* (see Table 3).

Booby Trap	No Bow	With Bow
Break-out	Time limit = 150	300
Snake pit	Penalty time = 75	100
Acid escape	Time = 500	1000

Table 3: Summary of booby trap buffs due to bow.

3.8 Gunpow[der] (Gg)

[;)] **(25 gems) Enables choice of booby trap minigame.** After crafting (and using) a unit of gunpowder, encountering a booby trap in the overworld presents a small menu, reproduced in Figure 10. Also allows explosive shots on the overworld, as shown in Figure 11. Each explosive arrow used requires **1 gem**, and will travel until it connects with sand, stone, the player, or the Great Pharaoh. Note: Gunpowder is **not available until bow has been crafted.**

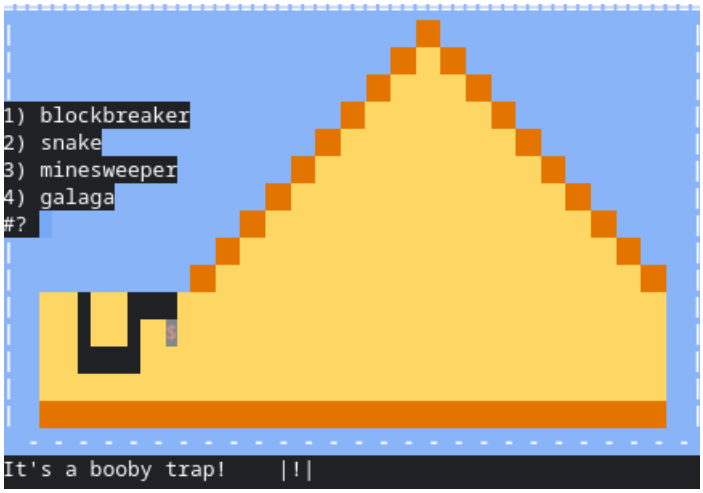


Figure 10: Reproduction of booby trap prompt, when equipped with explosive arrows. Entering 1-4 commences the associated minigame.

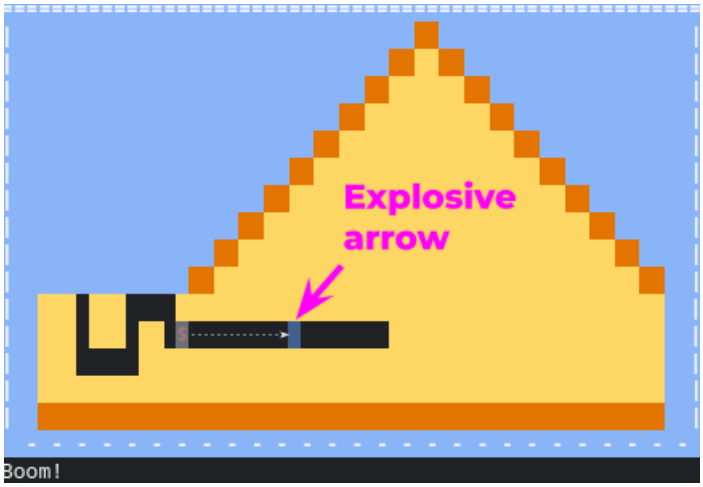


Figure 11: Visualization of explosive arrow, "mid-flight." Small, white, dotted arrow is drawn to guide the eye from player to shot. Upon contact with the right side of the tunnel, this shot destroys one block of sand, and whatever may lie beneath it. **The only thing an explosive shot *cannot* destroy is Pharaoh's sarcophagus!**

3.9 Spell (Ss)

[[]] **(5 or 50 gems)** Crumbling papyrus holds the hieroglyphic which **gifts the All-Seeing Eye**. When player has the Eye of Ra, the gods dispense a 90% discount in reverence of your fortitude.

4 Escaping with your Life

4.1 Wielding the All-Seeing Eye

To win, **the player must open the Great Pharaoh's sarcophagus before touching or destroying any other item on the map**. This is, of course, *incredibly* unlikely to happy accidentally, so the majority of successful games will utilize a lily and the All-Seeing Eye, in conjunction. Notice that destroying an item with an explosive arrow will also disqualify the player from opening the sarcophagus!

Figure 12: Example of overworld, after invoking the All-Seeing Eye. Inset figure displays correspondence between color and item.

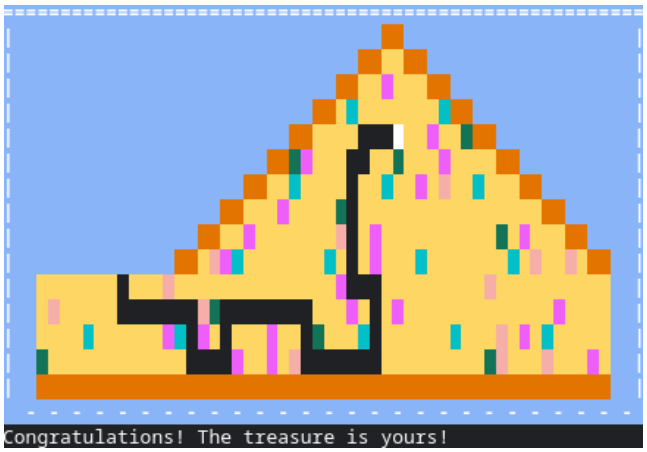


Figure 13: Valid solution of *Pharaoh's Fortune*, for the overworld given in Figure 12. Notice path does not disturb a single item, and terminates at Pharaoh's sarcophagus.

4.2 Final bonus

Simply opening Pharaoh's sarcophagus (worth 1,000,000 points!) may not be enough for the seasoned explorer. To put up high scores, you should remember:

$$\text{BONUS} = \text{ANKH}\cdot 10 + \text{AMULET}\cdot 10^2 + \text{DAGGER}\cdot 10^3 + \text{DAGGER}\cdot 10^4 + \text{LILY}\cdot 10^5 + \text{JARS}\cdot 10^6 + \text{BOW}\cdot 10^7 + \text{GUNPOW}\cdot 10^8 + \text{SPELL}\cdot 10^9$$

There will be no bonus if you are cursed when you touch the Pharaoh!

5 Tips, Tricks, & FAQ

$\mathcal{F}$eeling run-down? Unable to answer the mummies' questions? Tired of digging virtual sand? If so, this section of the guide cannot assist you in any way. If, however, you happen to have one of the very narrow inquiries listed below, I'm happy to help!

5.1 FAQ

Ummm, why won't the game start?
There might be many answers to that question. The first thing I'd check is that *Pharaoh's Fortune* is in your current working directory, with

```
ls | grep [Pp]har
```

If you don't see `phar_fo.sh`, or something similar, you may need to change directories, get the source file, etc.

Why is this game so awesome?
Glad you asked! Honestly, I'm not sure. Probably the multi-million dollar development team and corresponding production value. Feel free to sing your praises to rigzridge@gmail.com.

What's the best item?
It's hard to say. If you're cursed and down to one HP? Definitely craft ankhs. Just starting out, and feeling confident in your knowledge of ancient Egypt? I'd lean toward a dagger or jars. Are you looking to get an absurd score? Without question, choose spells. But to avoid only snarky responses, I'll offer this – when I play the game, the *first* item I craft is generally a dagger. Conceptually, I probably think the lily is the *best*, but the most pragmatic advice I can muster is *always keep an ankh and an amulet handy!*

Why is it faster to repeatedly smack the direction buttons than hold them down? I'm getting demolished at brickbreaker without glitching out on my J *and* L *keys.*
For better or worse, the pyramid conditions are what they are. That being said, if you have a good idea for the source code, email me and I'll grant access to the GitHub page. Assuming we bug-test everything, I'd be happy to merge your suggestions into the main branch!

Is walking the path of Ra a good thing?
I'm not sure about good, but it's definitely not a *safe* thing. On this path, your survival not only depends on how adept you are at booby traps and mummy questions, but also how *lucky* you are. Frankly, there are just no guarantees you'll find that magical sixth curse before dying in a terrible fashion. If, for example, you simply want to open the sarcophagus, then it's probably better to work through a couple of booby traps, answer a few questions, craft a thing or five, and buy a spell for 50 gems. If you do choose to walk the cursèd path, please do not forget that you must ward your curse or you will not recieve a bonus!

Okay, so being multiply cursed, fumbling around maniacally in booby traps doesn't really seem like my steez. What about the curse warder approach?
Great question! There's certainly a dichotomy here. If stacking curses is a gamble, then warding curses is a savings account. It doesn't accrue gems quickly, but it can be powerful for developing a sizable repository down the road (*i.e.*, playing the "long-game")! Warding curses can be fun, too, because it is intellectually rewarding to remove such maltempered magic.

What's the best way to win the game?
That's an important question, with an ocean of nuance behind it! Truly, I can't say for sure. What I can share is the *sneakiest* way I've found to win: Using an explosive arrow to open up the pyramid near Pharaoh!

6 Development Notes

Pharaoh's Fortune is incredibly cool to me. What began as an idea for a "neat" gift to accompany a Chromebook has blossomed into a project to be proud of (maybe?). If nothing else, it facilitated a much-needed crash-course in `bash` programming! Here's a history of my trials and tribulations:

```
# ------------------------------------------------------------
# 4/6/2022 -> Changed how visible radius is calculated with
# curses (now dx^2/4 + dy^2 < C(C+1), where C=# of curses)
# ------------------------------------------------------------
# 4/5/2022 -> Answers to mummies now converted to lower-case
# before parsing to eliminate "oSiRiS" type errors; now get
# a gem for warding curse (you should be incentivized to buy
# amulets _before_ you get cursed!); Eye of Ra needs 6 curses
# now (always thought 5 felt silly); dropped "Amun" from Eye
# of Amun-Ra (apparently it was *never* a thing); added a few
# new questions; changed version to 1.444 %^O
# ------------------------------------------------------------
# 4/3/2022 -> Added ability to stack amulets ("ward_stack" is
# new variable); changed 'curse_stack' to start counting from
# unity (initial curse now has "stack height" of 1)
# ------------------------------------------------------------
# 4/2/2022 -> Learned about the [[ ${arr[@]} "=~" "thing" ]]
# approach to finding duplicates in an array (was printf'ing
# it out then grep'ing); cleaned up questions and answers for
# mummy stuff (all in one readable array); much work pretty-
# ing up the help menu (3 pages in one array, better info)
# ------------------------------------------------------------
# 12/14/2021 -> Finally added "detailed stats"; hopefully can
# implement bow and gunpow buffs... Not looking good though
# ------------------------------------------------------------
# 11/28/2021 -> You now lose health if you fail blockbreaker
# before time's up; changed galaga bee color for contrast
# ------------------------------------------------------------
# 11/27/2021 -> Fixing this little stuff with eyeofHorus...
# Need to add a reset, my dude (not just quit && newgame!!!)
# ------------------------------------------------------------
# 11/23/2021 -> Slowed down galaga bees (they were moving on
# each frame); added gunpowder-driven minigame select; bonus
# only appears if not cursed; tut can now be one of 5 places;
# you lose health in minesweeper for each incorrect flag;
# mummy questions cleaned up a great deal (removed if...elif
# and started using arrays); galaga bees reduced from 16 to
# 12 for speed, now can fire up to 3 shots, display finished
# (readout of shots & bees), now check for collision 3x a loop
# (for shots) instead of 12x (for bees), shots continue even
# if bee is dead, print on either side of shot/ship/bee in
# attempt to curtail busy display; added cheat codes
# ------------------------------------------------------------
# 11/22/2021 -> Did some small clean-up of code; fixed issue
# with arrow taking extra half second (... || this && that ->
# ... || {this; that;}); curses now stack beyond 4 so vision
# can be huge (you die @ 10 curses though); fixed error that
# popped if high score name includes a number (just used '-m1'
# option for grep); bees in galaga now move as a row; want to
# make gunpowder give you choice of minigame, but 'select' has
# trouble printing the way I want; just realized you can use
```

```
# curse x 1 and lily to find tut, so I need to randomize him;
# galaga now only active when bow is purchased
# ------------------------------------------------------------
# 11/20/2021 -> Interested in CPU usage today... Averages are:
# Idling...8%, Menu...3%, Moving...40%(no wall)...90%(wall),
# Cursed...45%, Bow shot...35%, Brickbreak...75%, Snake...70%
# Minesweep...25%, Galaga...95%, Load minigame...100% %^/
# (hardware is Acer Spin311 Chromebook, so take it w/ a grain)
# Overall not wildly surprising, but bash is definitely not
# the best language for this. Portability is another issue
# I've been thinking about, so I might switch out 'echo's for
# 'printf's in every case pretty soon... Started working on
# that; added top 10 for high-scores; switched 'let n++' to
# ((n++)) [finally pulled the trigger!]; fixed color handling
# in blockbreaker to match that of galaga; changed means of
# randomization ('$RANDOM%2 && ...' -> '$RANDOM%1000>...) but
# I don't think it's better, removed it and called it a day!
# ------------------------------------------------------------
# 11/19/2021 -> Debugged the hell out of minesweeper, finally
# achieved full functionality, even caught a super-pernicious
# thing where 'num_arr' wasn't being cleared because we used
# 'local num_array=()', then changed the method of getting a
# 'Keypress' so time can count ('get_key'->'get_mini_control'),
# fixed arror in flagging system, added printing of endgame,
# swapped cogs of 'sweep_add_to_array_no_duplicate' for one
# based on 'grep' (simpler to understand!); cleaned up local
# vars in many functions, now using 'local -i' initialization;
# started on galaga minigame, essentially functional, have
# enemies that oscillate at different rates and shoot lasers
# randomly; honestly pretty umpressed by the progress here
# ------------------------------------------------------------
# 11/18/2021 -> Considered cleaning up right edge of display
# with 'tput ...' loop and adding "status bar", pretty bad
# idea (really, it's motivated by wanting to "fill the space"
# i.e., feature creep); added "death penance" where explosive
# arrow shot refills health and eliminates curse if player
# commits suicide while cursed %^}; gunpowder only available
# when bow is active; minigame loop in 'dig_find_animation'
# cleaned up; critical health (<=2) blinks red; learned about
# "((cond)) && do_if_cond" for replacing "if ((cond)); then
# do_if_cond; fi"; 'get_key' rewritten to handle arrow keys,
# but they are not recommended in minigames; fought a serious
# battle with proto-minesweeper, finally fixed it, sleepy...
# ------------------------------------------------------------
# 11/16/2021 -> Addressed jitter w/ minigame display by using
# 'tput' to keep printed text out of the way, also aligned
# display and changed "Game over!" string color to green;
# suppressed spurious "i,j,k,l" with 'printf "\e[8m"', which
# makes text hidden (use "\e[28m" to reset!); additional
# minor improvements (color, alignment, etc.)
# ------------------------------------------------------------
# 11/15/2021 -> Now +3 gems for mummy; "salt" exchanged for
# "gunpow", makes arrows explosive on main map, bow symbol
# in display changes when active, arrows hurt if shot around
# map; change to 'move_dude' that makes the dig result more
# modular (added 'dig_find_animation'); reduced curse radius;
```

```
# finally fucking realized it's "pharaoh" not "pharoah" =^/
# minigame section cleaned up a bit (moved 'print_mini' and
# 'get_mini_control' outside of 'minigame_snake'); added
# very rough version of blockbreaker, then slightly improved
# one; animation for trap is quicker now; new ending bonus
# ------------------------------------------------------------
# 11/14/2021 -> Fixed minigame negative health error (if you
# got to -1 health in minigame, you never died b/c game only
# checked for ==0); you now die when encountering a mummy with
# no dagger and 1 health; ring and jars finally functional;
# using '((n += cond?1:0))' instead of '((n = cond?n+1:n))';
# added bow consumable and tenative functionality, might
# switch with "salt" explosive
# ------------------------------------------------------------
# 11/10/2021 -> Color changes when snake eats eggs
# ------------------------------------------------------------
# 11/03/2021 -> Now prints symbol if dagger/ring/jars active;
# snake colors adjusted; curse stack improves item radius;
# more scoring available (e.g., per cursed step, length of
# snake, etc.); bonus multiplier if pharoah found; turned on
# zero blanking for score bar and adjusted y-intercept (<0)
# ------------------------------------------------------------
# 11/01/2021 -> Changed variable declarations (added local vars
# in subroutines and declared globals); curses now stack;
# printing of crafting menu is much more readable/efficient;
# dagger, ring, jars craftable (but not usable); various use
# of '[eval] printf "$str%.0s" {1..$limit}' for character reps;
# snake eggs change color, health bar active during minigame;
# cleaned 'draw_map' code [so many ((time--))s!], changed "time"
# to "step"; curse x 5 gets "Eye of Amun-Ra" for 1 gem
# ------------------------------------------------------------
# 10/31/2021 -> Added snake for booby traps
# ------------------------------------------------------------
# 10/30/2021 -> Working on bug fixes/code efficiency: map now
# reloads immediately after using a lily; crafting options added
# (dagger, ring, jars)
# ------------------------------------------------------------
# 10/29/2021 -> Fixed annoyance of "x" then "x -use" for using
# consumables; made single movement much faster by only drawing
# oldPos and currentPos (see draw_pyramid); updated help menu
# (page 2); *many* small improvements to code readability
# ------------------------------------------------------------
# 10/28/2021 -> Added subroutine to save code in crafting menu;
# using literate variables for colors; functionality for curses,
# mummies; new questions
# ------------------------------------------------------------
# 10/2?/2021 -> First code written
# ------------------------------------------------------------
```

7 Source Code

Because `bash` is just an interpreted shell language, it fails to provide a robust environment for game development. This fact alone should disqualify *Pharaoh's Fortune* from being a "good idea," but sometimes, it's nice to say to hell with sensible!

```
#!/bin/bash

############################################################
# Pharaoh's Fortune                        (c) 2021 G. Riggs
############################################################

phar_fo()
{
    # ------------------------
    # -=- Global variables -=-
    # ------------------------
    # [game fundamentals]
    local -i time=0 steps=0 gameOn=1

    # [map]
    local -i row=15 col=50 dum=5
    local -a map=() og_map=()
    local -i empty=-1 air=0 dude=1 rock=2 sand=3 king_tut=777 gem=666 mummy=555 curse=123 trap=911
    local -i cnt_gem cnt_curse cnt_trap cnt_mummy

    # [control flags]
    local -i weMovin=0 showStuff=0 needHelp=0 craftTable=0 askQuestion=0 resetMap=0 curseDefense=0 curseOn=0 miniGame=0
    local -i tutUnlocked=1 daggerActive=0 ringActive=0 jarsActive=0 bowActive=0 gunpowActive=0 eyeofHorus=0 eyeofRa

    # [consumables]
    local -i current_score=0 full_health=10 curse_stack=0 curse_steps=0 ward_stack=0
    local -i health=full_health jewels=0 ankh=0 amulet=0 dagger=0 ring=0 lily=0 jars=0 bow=0 gunpow=0 spell=0
    local item_list="ankh amulet dagger ring lily jars bow gunpow spell"

    # [final totals]
    local -i ankh_used=0 amulet_used=0 dagger_used=0 ring_used=0 lily_used=0 jars_used=0 spell_used=0
    local -i gem_found=0 curse_found=0 trap_found=0 mummy_found=0

    # [position]
    local -i currentPos oldPos

    # [minigames]
    local games=("blockbreaker" "snake" "minesweeper" "galaga")

    # [mummy questions]
    local numQuestions=20 questionNum=0
    local query question_arr=() questionList=$(eval echo {1..$numQuestions})
    for query in $(shuf -e $questionList)
    do
        question_arr[questionNum]=$((query-1))
        ((questionNum++))
    done
    questionNum=0
    unset query questionList

    # [user events]
    local Keypress

    # [colors]
    black_bg="\e[0m"
    bold_font="\e[1m"
    hidden_on="\e[8m"
    hidden_off="\e[28m"
    black_fg="\e[30m"
    red_fg="\e[31m"
    green_fg="\e[32m"
    yellow_fg="\e[33m"
    skyblue_fg="\e[34m"
    magenta_fg="\e[35m"
    cyan_fg="\e[36m"
    white_fg="\e[39m"
    red_bg="\e[41m"
    green_bg="\e[42m"
    dyellow_bg="\e[43m"
    skyblue_bg="\e[44m"
    magenta_bg="\e[45m"
    teal_bg="\e[46m"
    white_bg="\e[47m"
    pyellow_fg="\e[93m"
    gray_bg="\e[100m"
    salmon_bg="\e[101m"
    lyellow_bg="\e[103m"
    blink_cursor="\e[1;5;31;100m"

    # --------------------
    # -=- Introduction -=-
    # --------------------
    start_game()
    {
        local delay=0.05 cheatCode
        local -i i
        local strarr=("================================================="
                      "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-="
                      "   PPPP  H   H   A   RRRR    A    OOO  H   H  '   SSS   "
                      "   P   P H   H  A A  R   R  A A  O   O H   H  ' S      "
                      "   PPPP  HHHHH AAAAA RRRR  AAAAA O   O HHHHH      SSS   "
                      "   P     H   H A   A R  R  A   A O   O H   H         S  "
                      "   P     H   H A   A R   R A   A  OOO  H   H      SSS   "
                      "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-="
```

```
                "   FFFFF  OOO  RRRR  TTTTT U   U N   N EEEEE      /\\   "
                "   F     O   O R   R   T   U   U NN  N E         /  \\  "
                "   FFF   O   O RRRR    T   U   U N N N EEEE     /    \\ "
                "   F     O   O R  R    T   U   U N  NN E       /      \\"
                "   F      OOO  R   R   T    UUU  N   N EEEEE   -------- "
                "==--==--==--==--==--==--==--==--==--==--==--==--==--==--=="
                "=========================================================")
    clear
    printf "\n\n\n\n"
    for ((i=0; i<${#strarr[0]}; i++))
    do
        printf "${strarr[i]} \n"
        sleep $delay
    done
    printf "Loading:"
    for ((i=0; i<48; i++))
    do
        printf "X"
        sleep 0.01
    done
    printf "\n                [Press ENTER to begin]\n"
    read cheatCode
    case $cheatCode in
        ALLda_gems) jewels=666 ;;
        RaRaRa)     eyeofHorus=1 ;;
        deathwish)  curseOn=1; curse_stack=9 ;;
        OConnell)   bowActive=1; gunpowActive=1; ringActive=1; daggerActive=1; jarsActive=1; curseDefense=1; ward_stack=1; jewels=10 ;;
        khu-Foo)    col=65 ;;
        juicin!)    full_health=20; health=full_health ;;
        KB)         bowActive=1; gunpowActive=1; ringActive=1; daggerActive=1; jarsActive=1; curseDefense=0; ward_stack=0; jewels=999; eyeofHorus=1; spell=99;;
    esac
    local filename="PharFoLastAdventure.txt"
    local info_list="time steps gems curse_steps curse_stack gem_found trap_found curse_found mummy_found"
    printf "" > $filename
    for stat in $info_list $item_list
    do
        printf "%s " $stat >> $filename
    done
    printf "\n" >> $filename
}

# ---------------------
# -=- Initialize map -=-
# ---------------------
load_map()
{
    #set relevant flags
    cnt_gem=0 cnt_curse=0 cnt_trap=0 cnt_mummy=0
    currentPos=$((row-dum-1))
    weMovin=0 eyeofRa=0
    local -i mod=$((1 + $RANDOM%5))
    local tut_row_adjust=$((3-mod)) tut_col_adjust=$(($RANDOM%2))
    for ((i=0; i<row*col; i++))
    do
        if ((i%row-i/row/2+col/4+3==0 || (i%row+i/row/2==row & i%row<row-dum) || i%row==row-1)); then
            map[i]=$rock
        elif (((i%row-i/row/2+col/4+3>0 & i%row+i/row/2>row) || i%row>row-dum-1)); then
            if ((i==453+tut_col_adjust+row*tut_row_adjust)); then # Magic number in this line is just my favorite position for tut
                map[i]=$king_tut
            elif (( $RANDOM%2 & $RANDOM%2 & $RANDOM%2 & $RANDOM%2 )); then
                map[i]=$gem
                ((cnt_gem++))
            elif (( $RANDOM%2 & $RANDOM%2 & $RANDOM%2 & $RANDOM%2 & $RANDOM%2 )); then
                map[i]=$curse
                ((cnt_curse++))
            elif (( $RANDOM%2 & $RANDOM%2 & $RANDOM%2 & $RANDOM%2 & $RANDOM%2 )); then
                map[i]=$trap
                ((cnt_trap++))
            elif (( i%(22+mod)==0 )); then
                map[i]=$mummy
                ((cnt_mummy++))
            else
                map[i]=$sand
                #map[i]=$gem
            fi
        else
            map[i]=$air
        fi
        og_map[i]=${map[i]}
    done
    map[currentPos]=$dude
    printf "Legends speak of $cnt_gem gems scattered within the stone...\n"
    sleep 1
}

# -----------------
# -=- Get input -=-
# -----------------
get_key()
{
    read -sn 1 Keypress
    [ "$Keypress" == '' ] && { read -sn 1 Keypress; [ "$Keypress" == '[' ] && read -sn 1 Keypress; }
    # This is equivalent to
    #case $Keypress in
    #    $'\x1b')
    #        read -sn 1 Keypress
    #        case $Keypress in
    #            '[') read -sn 1 Keypress ;;
    #        esac ;;
    #esac
}

# ----------------------------
```

```
# -=- Switchyard for input -=-
# ----------------------------
move_dude()
{
    oldPos=$currentPos
    local -i currentRow=$((currentPos%row))
    local -i currentCol=$((currentPos/row))
    case $Keypress in
        [Jj]|D) ((currentCol+=col-1)) ;; # left
        [Kk]|B) ((currentRow+=1)) ;;     # down
        [Ll]|C) ((currentCol+=1)) ;;     # right
        [Ii]|A) ((currentRow+=row-1)) ;; # up
        [Oo]|'/') # shoot
            ((bowActive & gunpowActive & jewels>0)) && shoot_arrow || { printf "No arrows!\n"; sleep 0.5; } ;;
        [Qq]) # quit
            local sureQuit
            printf "Are you sure you want to quit? [y/n]\n"
            read -n 1 sureQuit && [ "$sureQuit" == "y" ] && gameOn=0 ;;
        [Zz]) # toggle Eye of Horus
            ((!eyeofHorus)) && { printf "You do not possess the all-seeing eye!\n"; sleep 1; } || ((showStuff=!showStuff)) ;;
        [Hh]) # help menu
            needHelp=1 ;;
        [Pp]|[Cc]) # pause/craft
            craftTable=1 ;;
        [Nn]|[Rr]) # New game/reset
            printf "Use [q] to quit current game first!\n"
            sleep 1 ;;
        *)
            printf "Unknown option... Use 'h' for help!\n"
            sleep 1
            weMovin=0;
            return ;;
    esac
    ((currentPos = currentRow%row + (currentCol%col)*row))
    local find
    for find in rock gem mummy curse trap king_tut
    do
        ((map[currentPos]==$find)) && { dig_find_animation $find; break; }
    done
    tput cup $((row + 6)) 0
    printf "                                                  \n"
    tput cup $((row + 6)) 0

    ((weMovin = oldPos!=currentPos?1:0))
    ((map[oldPos] = og_map[oldPos]>=sand?empty:0))
    ((current_score += map[currentPos]==sand?10:0))
    map[currentPos]=1
}

dig_find_animation()
{
    local -i i
    case $1 in
        rock)
            currentPos=$oldPos ;;
        gem)
            printf "Gem found!\n"
            for ((i=0; i<5; i++))
            do
                tput cup $((row + 6)) $((i+10))
                printf " --o"; sleep 0.05
            done
            printf "\b\b\b--|"; sleep 0.05; printf "\b\b\b -|"; sleep 0.05; printf "\b\b |"; sleep 0.05; printf "\b \n"
            ((gem_found++))
            ((jewels += curseOn&ringActive?3:1))
            ((current_score+=100)) ;;
        mummy)
            printf "Mummy encountered!  %%^o-/-<"
            for ((i=0; i<4; i++))
            do
                printf "\b\b\b\b\bo-|-<"; sleep 0.1; printf "\b\b\b\b\bu-/-<"; sleep 0.1;
                printf "\b\b\b\b\bu-|-<"; sleep 0.1; printf "\b\b\b\b\bu-\-<"; sleep 0.1
            done
            printf "\n"
            ((mummy_found++))
            ((health -= daggerActive?0:1))
            ((current_score+=500))
            askQuestion=1 ;;
        curse)
            printf "Curse detected..."
            for ((i=0; i<10; i++))
            do
                printf "\b\b\b.=^x"
                sleep 0.12
            done
            printf "\n"
            ((curse_found++))
            if ((curseDefense)); then
                curseOn=0
                eyeofRa=0
                ((ward_stack>1)) && { ((ward_stack--)); ((current_score+=curse_found*ward_stack*77)); ((jewels++)); } || curseDefense=0
            elif ((curseOn)); then
                ((curse_stack++))
                ((curse_stack==6)) && eyeofRa=1 # Turn on Eye of Ra
                ((curse_stack==10)) && gameOn=0 # You died
            else curseOn=1
                curse_stack=1
                curse_steps=0
            fi ;;
        trap)
            local bstr="\b\b\b\b\b\b\b"
            printf "It's a booby trap!  _w...w_" ; sleep 0.15; printf "${bstr}\w...w/"; sleep 0.15
            printf "${bstr}|w...w|"; sleep 0.15; printf "${bstr}|w...w|"; sleep 0.15
```

```
            printf "${bstr} |w..w|"; sleep 0.15; printf "${bstr} |w.w| "; sleep 0.15
            printf "${bstr}  |ww| "; sleep 0.15; printf "${bstr}  |!|  "; sleep 0.5
            ((trap_found++))
            weMovin=0
            local -i miniGame=1 game_draw=$(($RANDOM%(bowActive?4:3)))
            local game_choice=${games[game_draw]}
            if ((gunpowActive)); then
                local game
                tput cup 8 0
                select game in ${games[0]}
                do
                    case $game in
                        #"${games[0]/%/|}") game_choice=$game; break ;;
                        ${games[0]}|${games[1]}|${games[2]}|${games[3]}) game_choice=$game; break ;;
                    esac
                done
                draw_map 0
            fi
            clear_for_mini $game_choice
            eval minigame_${game_choice}
            draw_map 0 ;;
        king_tut)
            if ((tutUnlocked)); then
                printf "Congratulations! The treasure is yours!\n"
                local -i bonus_multiplier=1 bonus_score=0
                local item
                ((current_score+=1000000))
                for item in $item_list
                do
                    ((bonus_multiplier*=10))
                    ((bonus_score+=$item*bonus_multiplier))
                done
                ((curseOn==0)) && ((current_score+=bonus_score))
                printf "Bonus = $bonus_score\n"
                read
                gameOn=0
            else
                currentPos=$oldPos
                printf "...The ancient Pharaoh has awakened! The tomb is sealed!\n"
                printf "You must open the sarcophagus without touching anything!\n"
                printf "                [Press ENTER to continue]\n"
                ((health--))
                read
            fi
    esac
}

shoot_arrow()
{
    printf "Boom!\n"
    ((jewels--))
    local -i k=0 arrowFree=1 deathPenance=0 dumPos
    while ((arrowFree))
    do
        ((k++))
        dumPos=$((currentRow%row + ((currentCol+k)%col)*row))
        tput cup $((currentRow+5)) $(((currentCol+k)%col+3))
        if ((map[dumPos]==empty || map[dumPos]==air)); then
            arrowFree=1
        elif ((map[dumPos]!=king_tut)); then
            if ((map[dumPos]==dude)); then
                ((health--))
                if ((curseOn & health==0)); then
                    health=full_health
                    curseOn=0
                    deathPenance=1
                fi
            fi
            map[dumPos]=$empty
            arrowFree=0
        else break
        fi
        parse_space dumPos
        sleep 0.01
    done
    printf "$black_bg"
    if ((deathPenance)); then
        tput cup $((row + 6)) 0
        printf "You have absolved the ancient curse by giving your life.\n"
        printf "                [Press ENTER to continue]\n"
        read
    fi
}

# ------------------
# -=- Refresh map-=-
# ------------------
item_display_func()
{
    get_item_info $1
    eval local itemActive='$'$1Active
    (($itemActive)) && eval $1_str='\'"${cyan_fg}$sym'\'${white_fg}" || eval $1_str='\'"${black_fg}__'\'${white_fg}"
}

draw_map()
{
    # We need to clear the screen for a question else the skipped lines will be visible
    if (((weMovin & !askQuestion)||$1)); then
        tput cup 0 0;
    else clear; fi

    if ((curseOn & weMovin)); then
        ((curse_steps++))
```

```
        ((current_score+=13*curse_stack*curse_steps))
        local limited_stack=$((curse_stack<5?curse_stack:4))
        ((steps%(4-limited_stack+1)==0 )) && ((health--))
    fi

    if ((curseDefense)); then local curse_str="${cyan_fg}@x$((ward_stack))${white_fg}"
    elif ((curseOn)); then local curse_str="${red_fg}Cx$((curse_stack))${white_fg}"
    else curse_str="   "; fi

    if ((eyeofHorus)); then local eye_str="${green_fg}<(o)>${white_fg}"
    elif ((eyeofRa)); then local eye_str="${red_fg}<(o)>${white_fg}"
    else eye_str="     "; fi

    local item
    for item in dagger ring jars bow
    do
        item_display_func $item
    done
    ((bowActive&gunpowActive)) && bow_str="${cyan_fg}}${red_fg}>${white_fg}"

    ((health<=0)) && { gameOn=0; health=0; }
    ((health>full_health)) && health=full_health

    local dedbar="" livbar=""
    local -i i
    for ((i=0; i<health; i++)); do livbar="${livbar}|"; done
    for ((i=0; i<full_health-health; i++)); do dedbar="${dedbar} "; done

    if ((health<=2)); then crit_status="\e[5m"; else crit_status=""; fi

    printf "${bold_font}${white_fg}${skyblue_bg} Pharaoh's Fortune $eye_str                      [v1.444] \n"
    printf "=====================================================\n"
    printf " Steps:%4d  |  XPos = %2d   $jars_str $ring_str  Health:${curse_str}${red_bg}${crit_status}%s${green_bg}\e[25m%s${skyblue_bg} \n" $steps $((currentPos/row)) "$dedbar" "$livbar"
    printf " Gems x %3d  |  YPos = %2d   $bow_str $dagger_str  Score:  %12d \n" $jewels $((row-currentPos%row-6)) $current_score
    printf "=====================================================\n"
    if (($1)); then
        return
    elif ((needHelp||craftTable||askQuestion)); then
        if ((needHelp)); then
            run_help
            needHelp=0
        elif ((craftTable)); then
            craft_ish
        elif ((askQuestion)); then
            if ((health!=0)); then ask_question; fi
            printf "$black_bg"
            weMovin=0
            askQuestion=0
        fi
        draw_map 0
    elif ((weMovin & !curseOn)); then
        draw_pyramid $oldPos $currentPos
    else
        draw_pyramid -1 -1
    fi
}


# ----------------------
# -=- Crafting menu -=-
# ----------------------
get_item_info()
{
    case $1 in
        ankh)   sym="o+"; descrip="Boost health" ;;
        amulet) sym="@'"; descrip="Ward curses" ;;
        dagger) sym="?^"; descrip="Attack mummies" ;;
        ring)   sym="#0"; descrip="x3 gems if cursed" ;;
        bow)    sym="}-"; descrip="[Trap specific]" ;;
        lily)   sym="%*"; descrip="Refresh map" ;;
        jars)   sym="Oo"; descrip="HP+ from mummies" ;;
        gunpow) sym=";)"; descrip="Explosive arrows" ;;
        spell)
            sym="[]"
            ((eyeofRa)) && descrip="Eye of Ra" || descrip="Eye of Horus" ;;
    esac
}

print_craft_item()
{
    local thing=$1
    local first_letter="${thing: 0:1}" # Get first letter of string
    local sym=$2
    case $thing in amulet) first_letter="m" ;; lily) sym="%%*" ;;  esac  # Handle annoying cases
    eval local cost='$'$1_cost
    eval local num_of_thing='$'$thing

    local length_name=${#1}
    local length_descrip=${#3}
    dots_string=$(eval printf ".%.0s" {1..$((18-length_descrip))})

    local money_color=$red_fg
    ((jewels>=cost)) && money_color=$green_fg

    # This command is beefy...
    printf "| $sym [${magenta_fg}${first_letter^}${first_letter}${white_fg}] ${1^}:%*s${pyellow_fg}%2d${white_fg} ..... $3 $dots_string ${money_color}%2d${white_fg} gems |\n"  $((7-length_name)) " " $
}

craft_ish()
{
    local -i ankh_cost=1 amulet_cost=3 dagger_cost=5 ring_cost=7 lily_cost=8 jars_cost=10 bow_cost=15 gunpow_cost=25 spell_cost
    ((spell_cost = eyeofRa?5:50))
    local sym descrip item
```

```
    printf "${red_bg}|                    Crafting menu                     |\n"
    printf "|                  =-=-=-=-=-=-=-=                             |\n"
    printf "|    Item          #          Effect            Cost           |\n"
    printf "|   ------        ---        --------          ------          |\n"
    ((!bowActive)) && local item_list=$(printf "%s\n" $item_list | grep -v 'gunpow')
    for item in $item_list
    do
        get_item_info $item
        print_craft_item $item $sym "$descrip"
    done
    printf "|                                                              |%.0s\n" `eval echo {1..$((bowActive?1:2))}`
    printf "|    [Hh] Help                                                 |\n"
    printf "                  [Press ENTER to return]                      ${black_bg}\n"

    local -i moneyWarn=0 stuffWarn=0 loopCraft useIt
    local useItem
    read useItem
    local numBuy=$(echo $useItem | grep -o -E '[0-9]+')
    ((!numBuy>0)) && numBuy=1
    case $useItem in
      *-*[Uu]*) useIt=1 ;;
      *) useIt=0 ;;
    esac

    loopCraft=1
    case $useItem in
        [Aa]*)
            parse_menu_use ankh healthUp
            ((healthUp)) && { ((health+=numBuy)); healthUp=0; } ;;
        [Mm]*)
            parse_menu_use amulet curseDefense
            if ((curseDefense)); then
                ((curseOn)) && { curseOn=0; eyeofRa=0; curseDefense=0; ward_stack=0;} || ((ward_stack+=numBuy))
                ((ward_stack>9)) && ward_stack=9
            fi ;;
        [Dd]*)
            parse_menu_use dagger daggerActive ;;
        [Rr]*)
            parse_menu_use ring ringActive ;;
        [Ll]*)
            parse_menu_use lily resetMap
            eyeofHorus=0
            showStuff=0 ;;
        [Jj]*)
            parse_menu_use jars jarsActive ;;
        [Bb]*)
            parse_menu_use bow bowActive ;;
        [Gg]*)
            ((bowActive)) && parse_menu_use gunpow gunpowActive || printf "Not available without the bow!\n" ;;
        [Ss]*)
            parse_menu_use spell eyeofHorus
            ((eyeofHorus)) && showStuff=1 ;;
        [Hh]*)
            printf "See page 2 of help menu!" ;;
        '') craftTable=0 ;;
        *) printf "Unknown option!\n" ;;
    esac
    if ((moneyWarn)); then
        printf "Not enough gems...\n"; ((moneyWarn=0))
    elif ((stuffWarn)); then
        printf "None available...\n"; ((stuffWarn=0))
    fi
    ((craftTable)) && ((steps++))
    sleep 0.5
}

parse_menu_use()
{
    while ((loopCraft))
    do
        loopCraft=0
        if (($1>=$numBuy & useIt)); then
            (($1-=$numBuy))
            (($2=1))
            printf "$numBuy ${1^} used!\n"
            ((current_score+=$numBuy*$cost*10))
        else
            eval cost='$'${1}_cost
            if ((jewels>=$cost*$numBuy)); then
                (($1+=$numBuy))
                eval let ${1}_used+=$numBuy
                ((jewels-=$cost*$numBuy))
                get_item_info $1
                printf "Crafting $numBuy ${1^}:"
                echo -ne " $sym"
                local sleep_time=$(echo "scale=2; 0.05+$cost/100" | bc)
                local -i i
                for ((i=0; i<18; i++))
                do
                    echo -ne "\b\b- $sym"
                    sleep $sleep_time
                done
                printf "\n"
                ((useIt)) && loopCraft=1
            else
                moneyWarn=1
                stuffWarn=1
            fi
        fi
    done
}

# ----------------------
```

```
# -=- Refresh pyramid -=-
# ----------------------
draw_pyramid()
{
    if (($1<0)); then
        local -i i j k
        for ((j=0; j<row; j++))
        do
            printf "${skyblue_bg}|  "
            for ((k=0; k<col; k++))
            do
                ((i = j + row*k))
                parse_space i
            done
            printf "${white_fg}${skyblue_bg}  |\n"
        done
        printf "  - - - - - - - - - - - - - - - - - - - - - - - - - - - - ${black_bg}\n"
    else
        tput cup $((oldPos%row + 5)) $((oldPos/row + 3))
        parse_space oldPos
        tput cup $((currentPos%row + 5)) $((currentPos/row + 3))
        parse_space currentPos
        tput cup $((row + 6)) 0
    fi
}

parse_space()
{
    printf "$bold_font"
    if ((map[$1]==dude)); then
        printf "${blink_cursor}"
        ((curseOn)) && printf "!" || printf "$"
        printf "${black_bg}"
    elif ((map[$1]==empty)); then
        printf "${black_bg} "       # Excavated space
    elif ((map[$1]==rock)); then
        printf "${dyellow_bg} "     # Block
    elif ((map[$1]==sand)); then
        printf "${lyellow_bg} "     # Diggable
    elif ((map[$1]>sand & !showStuff & curseOn)); then
        local drow=$(((currentPos%row)-($1%row)))
        local dcol=$(((currentPos/row)-($1/row)))
        if ((drow*drow + dcol*dcol/4 < (1+curse_stack)*curse_stack)); then # This accounts for "squashiness" of display
            printf "${lyellow_bg}x" # Show collectible
        else
            printf "${lyellow_bg} " # Hide stuff
        fi
    elif ((map[$1]>sand & !showStuff)); then
        printf "${lyellow_bg} "     # Controls visibility
    elif ((map[$1]==gem)); then
        printf "${magenta_bg} "     # Amulet
    elif ((map[$1]==mummy)); then
        printf "${teal_bg} "        # Mummy
    elif ((map[$1]==curse)); then
        printf "${green_bg} "       # Bad juju
    elif ((map[$1]==trap)); then
        printf "${salmon_bg} "      # Map shift
    elif ((map[$1]==king_tut)); then
        printf "${white_bg} "       # Pharaoh
    else
        printf "${skyblue_bg} "     # Sky blue
    fi
}

# -----------------
# -=- Questions -=-
# -----------------
ask_question()
{
    local riddleAnswer
    printf "${red_bg}|                                                              |%.0s\n" 1
    printf "|                         Attack!                          |\n"
    printf "|                        =-=-=-=-=                         |\n"
    printf "|             Answer this riddle to pass...                |\n"
    printf "|                                                          |%.0s\n" {1..3}
    next_question
    printf "|                                                          |%.0s\n" {1..5}
    printf "   Think carefully! You only have one chance at this!   ${black_bg}\n"
    read riddleAnswer
    riddleAnswer=$(echo "$riddleAnswer" | tr '[A-Z]' '[a-z]') # Convert to lower-case
    case $riddleAnswer in
        $question_answer)
            printf "Yes! Gems earned!\n"
            ((jewels += 3))
            ((health += jarsActive?2:0))
            ((current_score += 1500)) ;;
        *) printf "Unfortunately not... "; ((health--))
    esac
    sleep 0.5
}

next_question()
{
    printf "${black_bg}\n"
    local qnum=${question_arr[questionNum]}
    ((questionNum++))
    ((questionNum==numQuestions)) && { questionNum=0; ((current_score+=100000)); }

    local questionList=("[             What is the pharaoh's name?            ]"  # 0
                        "*[Tt]ut*" # NOT TUT! is best answer
                        "[      Which god posessed the head of a jackal?       ]"  # 1
                        "*[Aa]nubis*"
                        "[      What will your heart be weighed against?       ]"  # 2
```

```
                    "*[Ff]eather*"
                    "[                  Who is Set's brother?              ]"  # 3
                    "*[Oo]siris*"
                    "[     Is Upper Egypt north or south of Lower Egypt?   ]"  # 4
                    "*[Ss]outh*"
                    "[         Which god has the least understood head?    ]"  # 5
                    "*[Ss]et*"
                    "[          Which direction does Ra go to die?         ]"  # 6
                    "*[Ww]est*"
                    "[                Whose floods bring life?             ]"  # 7
                    "*[Nn]ile*"
                    "[        Where does Ra battle through the night?      ]"  # 8
                    "*[Dd]uat*"
                    "[                  Who rides the Mandjet?             ]"  # 9
                    "*[Rr]a*"
                    "[                Which god is ithyphallic?            ]"  # 10
                    "*[Mm]in*"
                    "[                Who is goddess of the sky?           ]"  # 11
                    "*[Nn]ut*"
                    "[                  Isis begot which god?              ]"  # 12
                    "*[Hh]orus*"
                    "[          Sekhmet is depicted as what animal?        ]"  # 13
                    "*[Ll]ion*"
                    "[          Who represents the primordial ocean?       ]"  # 14
                    "*[Nn]u*"
                    "[       Who emerged from Nu, sitting on a benben?     ]"  # 15
                    "*[Aa]tum*"
                    "[             How did Atum create the world?          ]"  # 16
                    "*[Mm]asturbat*"
                    "[   Who has green skin and is betrothed to a lioness? ]"  # 17
                    "*[Pp]tah*"
                    "[                  What is Sobek's head?              ]"  # 18
                    "*[Cc]roc*"
                    "[               What did Akhenaten worship?           ]"  # 19
                    "*[Aa]ten*")

    printf "${questionList[2*qnum]}\n"
    question_answer=${questionList[2*qnum+1]}
    printf "${bold_font}${red_bg}\n"
}

# -----------------
# -=- Check map -=-
# -----------------
count_map()
{
    local -i i chk_gem=0 chk_curse=0 chk_trap=0 chk_mummy=0
    tutUnlocked=0
    for ((i=0; i<row*col; i++))
    do
        if ((map[i]==gem)); then
            ((chk_gem++))
        elif ((map[i]==curse)); then
            ((chk_curse++))
        elif ((map[i]==trap)); then
            ((chk_trap++))
        elif ((map[i]==mummy)); then
            ((chk_mummy++))
        fi
    done
    ((cnt_gem==chk_gem & cnt_curse==chk_curse & cnt_trap==chk_trap & cnt_mummy==chk_mummy)) && tutUnlocked=1
}

# -----------------
# -=- Help menu -=-
# -----------------
run_help()
{
    local help_menu=("${red_bg}                      Help menu                       "
                     "${black_bg}|                      =-=-=-=-=                       |"
                     "| Welcome to Pharaoh's Fortune! Your goal is to discov-|"
                     "| er the secret of the sacred pyramid. Before unlocking|"
                     "| Pharaoh's sarcophagus, you will need to collect ${magenta_fg}gems${white_fg} |"
                     "| and solve riddles, all while avoiding the scourge of |"
                     "| undead ${cyan_fg}mummies${white_fg}, ${pyellow_fg}booby traps${white_fg}, and ${green_fg}curses${white_fg}... Good luck!|"
                     "|                                                      |"
                     "|                  ---- Controls ----                  |"
                     "|       i/j/k/l -> move up/left/down/right             |"
                     "|       p/c     -> pause/crafting menu                 |"
                     "|       h       -> this menu                           |"
                     "|       q       -> quit                                |"
                     "|       z       -> toggle all-seeing eye (${red_fg}if available${white_fg})|"
                     "|       o       -> shoot explosive arrow (${red_fg}if available${white_fg})|"
                     "${black_bg}${red_bg}                ${black_fg}[Press ENTER to continue]          (1/3)${black_bg}"
                     "|                  --- Crafting ---                    |"
                     "|   Input [Xx] of item then [ENTER] to buy. Use items  |"
                     "|   with '-u' or '-use' directive. Will automatically  |"
                     "|   craft item if requested amount exceeds inventory.  |"
                     "|   {Ex1: 's 3' or 'spell 3' buys 3 spells}            |"
                     "|   {Ex2: 'a -u 9' or 'ankh -use 9' uses 9 ankhs}      |"
                     "|                                                      |"
                     "|              ---- Eye of Horus ----                 |"
                     "|          (Use to see beneath the sand!)             |"
                     "|         ${magenta_bg}   ${black_bg}     = Gem                              |"
                     "|         ${teal_bg}   ${black_bg}     = Mummy                           |"
                     "|         ${salmon_bg}   ${black_bg}     = Trap                             |"
                     "|         ${green_bg}   ${black_bg}     = Curse                           |"
                     "|         ${white_bg}   ${black_bg}     = The Great Pharaoh               |"
                     "${red_bg}                ${black_fg}[Press ENTER to continue]          (2/3)${black_bg}"
                     "|                --- Inventory 101 ---                 |"
                     "| These items will improve your chances of survival... |"
                     "| Use them as quickly as possible!                     |"
                     "| ${yellow_fg}Dagger${white_fg} [ ${cyan_fg}?^${white_fg} ]                                      |"
```

```
                    "| Protects against initial mummy attack               |"
                    "|                                                     |"
                    "| ${yellow_fg}Jars${white_fg}    [ ${cyan_fg}Oo${white_fg} ]                                    |"
                    "| +2 HP when mummy's question is answered correctly   |"
                    "|                                                     |"
                    "| ${yellow_fg}Ring${white_fg}    [ ${cyan_fg}#O${white_fg} ]                                    |"
                    "| Gems found in overworld will be tripled while cursed |"
                    "|                                                     |"
                    "| ${yellow_fg}Bow${white_fg}     [ ${cyan_fg}}-${white_fg} ]  +  ${green_fg}Gunpow${white_fg}    [ ${cyan_fg}}${red_fg}>${white_fg} ]                    |"
                    "| All booby traps accessible; gunpowder enables shot  |"
                    #"|                                                     /"
                    "${red_bg}                ${black_fg}[Press ENTER to return]           (3/3)${black_bg}")
    local -i i=0
    printf "%b\n" "${help_menu[i]}"
    for ((i=1; i<${#help_menu[@]}; i++))
    do
        printf "%b\n" "${help_menu[i]}"
        ((i%15 == 0)) && { read; tput cup 6 0; }
    done
}

# -----------------
# -=- Minigames -=-
# -----------------
clear_for_mini()
{
    tput cup 12 16
    case $1 in
        blockbreaker) printf "You are trapped by bricks!\n" ;;
        snake)        printf "You fell into a snake pit!\n" ;;
        minesweeper)  printf "Beware the vile salt acid!\n" ;;
        galaga)       printf "Ancient ETs are attacking!\n" ;;
    esac
    local -i i
    for ((i=0; i<row*col; i++))
    do
        print_mini $((i%row)) $((i/row)) "${skyblue_bg} "
    done
}

print_mini()
{
    tput cup $(($1 + 5)) $(($2 + 3))
    printf "$3"
}

get_mini_control()
{
    tput cup $((row + 7)) 0
    read -sn 1 -t 0.01 Keypress
    sleep $mini_sleep_time
    case "$Keypress" in
        [Jj]|D)   mini_flag="left"   ;;
        [Ll]|C)   mini_flag="right"  ;;
        [Ii]|A)   mini_flag="up"     ;;
        [Kk]|B)   mini_flag="down"   ;;
        [Oo]|'/') mini_flag="shoot"  ;;
        [Zz])     mini_flag="reveal" ;;
        *)        mini_flag="none"   ;;
    esac
}

minigame_blockbreaker()
{
    local -i cnt=0 block_score=0
    local -i mini_time_limit=$((bowActive?300:150))
    local block_arr=() block_hue=("$teal_bg" "$red_bg" "$magenta_bg" "$white_bg")

    local -i block_row=$((row-1)) block_col=15
    local -i old_block_col
    local mini_flag="right"

    local -i ball_row=$((block_row/2)) ball_col=$((block_col/2))
    local ball_x=$ball_col
    local ball_velx=1 ball_vely=-1

    local mini_sleep_time=0.05
    local block_time_frac block_strA block_strB
    local -i length_block_string=21

    block_initialize()
    {
        local -i i j k
        for ((j=0; j<4; j++))
        do
            for ((k=0; k<col; k+=6))
            do
                ((i = j + k*row))
                block_arr[i]=$((4-j))
                print_mini $j $k "${block_hue[j]} "
            done
        done
    }

    block_move_paddle()
    {
        old_block_col=$block_col
        case $1 in
            left)  ((block_col+=col-2));;
            right) ((block_col+=2));;
        esac
        ((block_col%=col))
```

```
        print_mini $block_row $((old_block_col%col-3)) "${skyblue_bg}       "
        print_mini $block_row $((block_col%col-3))     "${dyellow_bg}       "
        mini_flag="none"
    }

    block_move_ball()
    {
        print_mini $ball_row $ball_col "${skyblue_bg} "

        ball_x=$(echo "scale=2; $ball_x + 0.25*$ball_velx" | bc )
        ball_col=$(echo "scale=0; $ball_x/1" | bc )

        ((ball_row+=ball_vely))

        ((ball_col<=1 || ball_col>=col-2)) && ((ball_velx*=-1))

        if ((ball_row==0)); then
            ((ball_vely*=-1))
        elif ((ball_row<=3)); then
            local -i ballPos=$((ball_row + $ball_col*row))
            if ((block_arr[ballPos]!=0)); then
                ((ball_vely*=-1))
                ((block_arr[ballPos]==1)) && ((health--))
                ((block_score+=block_arr[ballPos]))
                block_arr[$ballPos]=0
            fi
        elif ((ball_row==block_row)); then
            local -i val=$((ball_col-block_col))
            local -i abs_val=$(echo $val | grep -o -E '[0-9]+')
            if ((abs_val<=4)); then
                ((ball_velx+=val))
                ((ball_vely*=-1))
            else miniGame=0
                ((health-=gunpowActive?0:1))
                return
            fi
        fi

        print_mini $ball_row $ball_col "${lyellow_bg} "
    }

    block_initialize
    while ((miniGame & cnt<=mini_time_limit))
    do
        printf "$hidden_on"
        block_move_ball
        get_mini_control
        printf "$hidden_off"

        tput cup $((row + 6)) 0
        block_time_frac=$(((cnt*length_block_string)/mini_time_limit))
        block_strA=$(eval printf "X%.0s" {1..$((length_block_string-block_time_frac))})
        block_strB=$(eval printf ".%.0s" {1..$block_time_frac})

        printf "${green_bg}| Time remaining:: %s%s | Score::%2d |\n" $block_strA $block_strB $block_score

        block_move_paddle $mini_flag
        draw_map 1
        ((cnt++))
    done
    ((jewels+=block_score))
    ((current_score+=cnt*block_score))
    tput cup $((row + 6)) 0
    printf "${dyellow_bg} Game over! You collected %2d mudbricks!                \n" $block_score
    read
    print_mini $((row + 1)) -3 "                                                                  $black_bg"
}

minigame_snake()
{
    local -i init_row=2 init_col=3
    local -i snake_pos=$((init_row + init_col*row))

    local -i max_length=30 snake_length=4 snake_tail=0 snake_head=1

    local -i isEat=0 snake_food=-1
    local mini_flag="right" old_snake_flag
    local snake_arr=()

    local -i cnt=0 apples=0 food_time=0
    local mini_sleep_time=0.07
    local -i mini_time_limit=500
    local -i snake_time_penalty=$((bowActive?100:75))

    make_snake_food()
    {
        ((snake_food<0)) && snake_food=155 || snake_food=$(($RANDOM%(row*col)))
        print_mini $((snake_food%row)) $((snake_food/row)) "${red_bg} "
    }

    snake_initialize()
    {
        local -i i
        for ((i=0; i<max_length; i++))
        do
            snake_arr[i]=-1
        done

        snake_arr[0]=$((snake_pos+1))  # Tail
        snake_arr[1]=$snake_pos # Head
        make_snake_food
        print_mini $((snake_pos%row)) $((snake_pos/row)) "${green_bg} "
    }
```

```
    check_snake_collision()
    {
        local dum=$1
        snake_collide=0
        local -i i
        for ((i=snake_tail; i<snake_tail+snake_length-1; i++))
        do
            if ((snake_arr[i%max_length]==dum)); then
                snake_collide=1
                return
            fi
        done
    }

    move_snake_once()
    {
        ((snake_row=snake_pos%row))
        ((snake_col=snake_pos/row))
        case $1 in
            left)  ((snake_col+=col-1)) ;;
            right) ((snake_col++)) ;;
            up)    ((snake_row+=row-1)) ;;
            down)  ((snake_row++)) ;;
        esac

        ((snake_tail_pos = snake_arr[snake_tail]))
        print_mini $((snake_tail_pos%row)) $((snake_tail_pos/row)) "${skyblue_bg} "

        ((snake_tail = isEat?snake_tail:(snake_tail+1)%max_length))
        isEat=0

        print_mini $((snake_pos%row)) $((snake_pos/row)) "${lyellow_bg} "
        ((snake_head = (snake_tail+snake_length-1)%max_length))
        ((snake_pos = snake_row%row + (snake_col%col)*row))

        ((snake_arr[snake_head] = snake_pos))
        check_snake_collision $snake_pos
        if ((snake_collide)); then
            miniGame=0
            ((health-=gunpowActive?0:1))
        fi
        print_mini $((snake_pos%row)) $((snake_pos/row)) "${green_bg} "

        if (((cnt-food_time)%snake_time_penalty==0 & cnt!=0)); then
            print_mini $((snake_food%row)) $((snake_food/row)) "${magenta_bg} "
            ((health--))
        fi

        if ((snake_pos==snake_food)); then
            ((apples++))
            mini_sleep_time=$(echo "scale=2; $mini_sleep_time*0.95" | bc)
            isEat=1
            ((snake_length+=2))
            snake_collide=1
            while ((snake_collide));
            do
                make_snake_food
                check_snake_collision $snake_food
            done
            food_time=$cnt
        fi
    }

    snake_initialize
    while ((miniGame & cnt<=mini_time_limit))
    do
        printf "$hidden_on"
        old_snake_flag=$mini_flag
        get_mini_control
        case $mini_flag in
            none) mini_flag=$old_snake_flag ;;
        esac
        move_snake_once $mini_flag
        check_snake_collision
        printf "$hidden_off"

        tput cup $((row+6)) 0
        ((!isEat)) && score_color="$green_bg" || score_color="$red_bg"
        printf "${score_color}| Time remaining::%4d | Snake eggs::%3d | Split::%4d |\n" $((mini_time_limit-cnt)) $apples $((cnt-food_time))

        draw_map 1
        ((cnt++))
    done
    ((jewels+=apples))
    ((current_score+=apples*snake_length*10))
    tput cup $((row+6)) 0
    printf "${lyellow_bg} Game over! You collected %2d snake eggs!                \n" $apples
    read
    print_mini $((row+1)) -3 "                                                                $black_bg"
}

minigame_minesweeper()
{
    local -i cnt=0 sweep_score=0 mini_time_limit=$((bowActive?1000:500))
    local mini_sleep_time=0.15

    local -i sweep_offsetr=$((row/4)) sweep_offsetc=$((col/4))
    local -i sweep_row=$((row - 2*sweep_offsetr)) sweep_col=$((col - 2*sweep_offsetc))
    local -i sweep_posr=$((sweep_row/2)) sweep_posc=$((sweep_col/2))

    local mine_map=() sweep_map=() sweep_updated_map=() sweep_flags=()
```

```
local sweep_color=( " " "${white_fg}1" "${green_fg}2" "${pyellow_fg}3" "${cyan_fg}4" "${yellow_fg}5" "${yellow_fg}6" "${yellow_fg}7" "${yellow_fg}8" "${red_fg}*")
local -i mine_num=0 flag_num=0 right_num=0

local sweep_bg="\e[48;5;249m"
local initial_space="${bold_font}${black_fg}#"

sweep_func()
{
    local -i i j k
    for ((j=0; j<sweep_row; j++))
    do
        for ((k=0; k<sweep_col; k++))
        do
            ((i = j + k*sweep_row))

            $1
        done
    done
}

sweep_local_func()
{
    local -i ii jj kk
    for ((jj=-1; jj<=1; jj++))
    do
        for ((kk=-1; kk<=1; kk++))
        do
            if ((j+jj>=0 & j+jj<sweep_row & k+kk>=0 & k+kk<sweep_col)); then
                ((ii = j+jj + (k+kk)*sweep_row))

                $1
            fi
        done
    done
}

sweep_initialize()
{
    if (( $RANDOM%2 & $RANDOM%2 & $RANDOM%2 )); then
        ((mine_num++))
        mine_map[i]=1
    else
        mine_map[i]=0
    fi
    sweep_updated_map[i]=$initial_space
    print_mini $((j+sweep_offsetr)) $((k+sweep_offsetc)) "${sweep_bg}${sweep_updated_map[i]}"
}

sweep_add_mine_count()
{
    ((chk_mine_num += mine_map[ii]))
}

sweep_check_neighbors()
{
    local chk_mine_num=0
    if ((mine_map[i]==1)); then
        chk_mine_num=9
    else
        sweep_local_func sweep_add_mine_count
    fi
    sweep_map[i]=$chk_mine_num
}

sweep_prune_array_of_element()
{
    local -n dum_arr=$1
    local -i n dum_num=${#dum_arr[@]}
    for ((n=0; n<dum_num; n++))
    do
        ((dum_arr[n]==$2)) && unset dum_arr[n]
    done
}

sweep_move()
{
    local i=$((sweep_posr + $sweep_posc*sweep_row))
    print_mini $((sweep_posr+sweep_offsetr)) $((sweep_posc+sweep_offsetc)) "${sweep_bg}${sweep_updated_map[i]}${black_fg}\b"
    case $1 in
        left)  ((sweep_posc -= sweep_posc==0?0:1));;
        right) ((sweep_posc += sweep_posc==sweep_col-1?0:1));;
        up)    ((sweep_posr -= sweep_posr==0?0:1));;
        down)  ((sweep_posr += sweep_posr==sweep_row-1?0:1));;
        shoot)
            case ${sweep_updated_map[i]} in
                "${red_fg}?") sweep_updated_map[i]=$initial_space;;
                "${red_fg}F")
                    sweep_updated_map[i]="${red_fg}?"
                    sweep_prune_array_of_element sweep_flags $i ;;
                *)
                    sweep_updated_map[i]="${red_fg}F"
                    sweep_flags+=($i) ;;
            esac
            flag_num=${#sweep_flags[@]} ;;
        reveal)
            if ((sweep_map[i]==0)); then
                sweep_clear_board
            elif ((sweep_map[i]==9)); then
                miniGame=0
                ((health-=gunpowActive?0:1))
            else sweep_updated_map[i]=${sweep_color[sweep_map[i]]}
            fi ;;
    esac
```

```
        local sweep_cursor_pos=$((sweep_posr + sweep_posc*sweep_row))

        print_mini $((sweep_posr+sweep_offsetr)) $((sweep_posc+sweep_offsetc)) "${red_bg}${sweep_updated_map[$sweep_cursor_pos]}${sweep_bg}${black_fg}\b"
    }

    sweep_clear_board()
    {
        local -i n cur_ind=1
        local blank_arr=() num_arr=()
        blank_arr[cur_ind]=$i

        while ((cur_ind<=${#blank_arr[@]}))
        do
            sweep_get_blanks ${blank_arr[cur_ind]}
            ((cur_ind++))
        done

        blank_arr+=(${num_arr[@]})

        for ((n=1; n<=${#blank_arr[@]}; n++))
        do
            sweep_updated_map[blank_arr[n]]=${sweep_color[sweep_map[blank_arr[n]]]}
            print_mini $((blank_arr[n]%sweep_row+sweep_offsetr)) $((blank_arr[n]/sweep_row+sweep_offsetc)) "${sweep_updated_map[blank_arr[n]]}"
        done
    }

    sweep_get_blanks()
    {
        local -i j=$(($1%sweep_row)) k=$(($1/sweep_row))
        sweep_local_func sweep_check_blank_and_such
    }

    sweep_check_blank_and_such()
    {
        if ((sweep_map[ii]==0)); then
            sweep_add_to_array_no_duplicate blank_arr $ii
        elif ((sweep_map[ii]<9)); then
            sweep_add_to_array_no_duplicate num_arr $ii
        fi
    }

    sweep_add_to_array_no_duplicate()
    {
        local -n dum_arr=$1
        isDuplicate=0
        [[ ${dum_arr[@]} =~ "$2" ]] && isDuplicate=1 || dum_arr+=($2)
        # This is equivalent to:
        #if printf '%s\n' "${dum_arr[@]}" | grep -q "^$2$"; then
        #    isDuplicate=1
        #else
        #    dum_arr+=($2)
        #fi
    }

    sweep_end_game()
    {
        local -i i n pos
        for ((i=0; i<sweep_row*sweep_col; i++))
        do
            ((sweep_map[i]==9)) && print_mini $((i%sweep_row+sweep_offsetr)) $((i/sweep_row+sweep_offsetc)) "${red_fg}*"
        done
        for ((n=0; n<flag_num; n++))
        do
            pos=${sweep_flags[n]}
            if ((sweep_map[pos]==9)); then
                ((sweep_score++))
                print_mini $((pos%sweep_row+sweep_offsetr)) $((pos/sweep_row+sweep_offsetc)) "${magenta_fg}@"
            else
                ((health--))
                print_mini $((pos%sweep_row+sweep_offsetr)) $((pos/sweep_row+sweep_offsetc)) "${red_fg}X"
            fi
        done
    }

    sweep_func sweep_initialize
    sweep_func sweep_check_neighbors && unset mine_map
    while ((miniGame & cnt<=mini_time_limit))
    do
        get_mini_control
        sweep_move $mini_flag

        tput cup $((row+6)) 0
        printf "${white_bg}| Time remaining:: %s | Found:: %2d | Total vats:: %2d |\n${sweep_bg}" $((mini_time_limit-cnt)) $flag_num $mine_num

        ((cnt++))
    done
    sweep_end_game
    ((jewels+=sweep_score))
    ((current_score+=cnt*sweep_score))
    tput cup $((row+6)) 0
    printf "${gray_bg}${white_fg} Game over! You uncovered %2d vats of acid!              \n" $sweep_score
    read
    print_mini $((row+1)) -3 "                                                              $black_bg"
}

minigame_galaga()
{
    local -i cnt=0 galaga_score=0 num_aliens
    local -i mini_time_limit=300
    local alien_arr=() bees_shot=() bees=() shots=(0 0 0)

    local -i ship_row=$((row-2)) ship_col=15
    local mini_flag="none"
```

```
local shot_row=(0 0 0) shot_col=(0 0 0)
local shot_x=$shot_col
local shot_velx=1 shot_vely=-1

local mini_sleep_time=0.01
local bee_hue=("$green_bg" "$red_bg" "$teal_bg" "$lyellow_bg")
local bees_str shot_str

galaga_initialize()
{
    local -i i j k n=0
    for ((j=2; j<=8; j+=2))
    do
        for ((k=10; k<col; k+=15))
        do
            ((i = j + k*row))
            ((n++))
            alien_arr[n]=$i
            bees_shot[n]=0
            print_mini $((i%row)) $((i/row)) "${bee_hue[$((j/2-1))]} "
        done
    done
    num_aliens=$n
    bees=($((n/4)) $((n/4)) $((n/4)) $((n/4)))
}

galaga_move_ship()
{
    print_mini $ship_row $((ship_col-1)) "${skyblue_bg}   "
    case $1 in
        left)  ((ship_col -= ship_col==0?0:1))     ;;
        right) ((ship_col += ship_col==col-1?0:1)) ;;
        up)    ((ship_row -= ship_row==row-5?0:1)) ;;
        down)  ((ship_row += ship_row==row-2?0:1)) ;;
        shoot|reveal)
            local shots_now=$((${shots[@]/%/+}0)) # This is great! First, we append a "+" to each element, then evaluate in (())!
            if ((shots_now!=3)); then             # ...probably the coolest way to sum an array!
                local which_shot=$(printf "%s\n" ${shots[@]} | grep -m1 -n "0") # Find first zero and get index
                which_shot=$((${which_shot%:*}-1))
                if ((which_shot!=-1)); then
                    shot_row[which_shot]=$ship_row
                    shot_col[which_shot]=$ship_col
                    shots[which_shot]=1
                fi
            fi ;;
    esac
    print_mini $ship_row $ship_col "${gray_bg}^"
    mini_flag="none"
}

galaga_move_shot()
{
    local -i n shotPos
    local which_bee
    for n in {0..2}
    do
        if ((shots[n]!=0)); then
            print_mini ${shot_row[n]} $((shot_col[n]-1)) "${skyblue_bg}   "

            ((shot_row[n]--))

            if ((shot_row[n]<0)); then
                shots[n]=0
            else
                shotPos=$((shot_row[n] + shot_col[n]*row))

                which_bee=$(printf "%s\n" ${alien_arr[@]} | grep -m1 -n "^$shotPos$") # Find if collision and where!
                which_bee=${which_bee%:*}                                             # Peel off everything behind ":"

                if ((which_bee!=0)); then
                    alien_arr[which_bee]=0
                    ((galaga_score+=5-shot_row[n]/2))
                    ((bees[$((shot_row[n]/2-1))]--))
                    print_mini ${shot_row[n]} $((shot_col[n]-1)) "${skyblue_bg}   "
                    shots[n]=0
                fi

                ((shots[n])) && print_mini ${shot_row[n]} ${shot_col[n]} "${white_bg} "
            fi
        fi
    done
}

galaga_move_bees_and_shots()
{
    local -i i u n k dumrow dumcol dumdir
    local dumhue dumdir dumdir_arr=(0 0 0 0)
    for k in {0..3}
    do
        (($RANDOM%4==0)) && dumdir_arr[k]=$(($RANDOM%3 - 1))
    done
    for ((n=1; n<=num_aliens; n++))
    do

        if ((bees_shot[n]!=0)); then

            u=${bees_shot[n]}
            print_mini $((u%row)) $((u/row - 1)) "${skyblue_bg}   "
            if ((u-1==ship_row+ship_col*row || u==ship_row+ship_col*row)); then
                miniGame=0
                ((health-=gunpowActive?0:1))
            elif ((u%row!=row-1)); then
```

```
                print_mini $((u%row+1)) $((u/row)) "${magenta_bg} "
                bees_shot[n]=$((u+1))
            else
                bees_shot[n]=0
            fi
        fi

        if ((alien_arr[n]!=0)); then

            dumrow=$((alien_arr[n]%row))
            dumcol=$((alien_arr[n]/row))
            print_mini $dumrow $((dumcol-1)) "${skyblue_bg}   "

            dumdir=${dumdir_arr[$((dumrow/2-1))]}
            ((dumcol+=dumdir<0?col-1:dumdir))
            ((dumcol%=col))

            dumhue=${bee_hue[$((dumrow/2-1))]}

            i=$((dumrow + dumcol*row))
            alien_arr[n]=$i
            print_mini $dumrow $dumcol "${dumhue} "
            ((bees_shot[n]==0)) && (($RANDOM%100>95)) && bees_shot[n]=$i
        fi

    done
}


galaga_initialize
while ((miniGame & cnt<=mini_time_limit))
do
    printf "$hidden_on"
    galaga_move_ship $mini_flag
    get_mini_control
    galaga_move_shot
    galaga_move_bees_and_shots
    printf "$hidden_off"

    shot_str=""
    for n in {0..2}
    do
        ((shots[n])) && shot_str="${shot_str}${green_bg}  " || shot_str="${shot_str}${white_bg} ${green_bg} "
    done
    bees_str="${bee_hue[3]}${bees[3]} ${bee_hue[2]}${bees[2]} ${bee_hue[1]}${bees[1]} ${bee_hue[0]}${bees[0]}"
    tput cup $((row+6)) 0
    printf "${black_bg}${green_bg}| Time:: %s | Score::%2d | Shots:: ${shot_str}| ETs::${bees_str} \n" $((mini_time_limit-cnt)) $galaga_score

    ((cnt++))
done
((jewels+=galaga_score))
((current_score+=cnt*galaga_score))
tput cup $((row+6)) 0
printf "${dyellow_bg} Game over! You destroyed %2d alien invaders!            \n" $galaga_score
read
print_mini $((row+1)) -3 "                                                              $black_bg"
}

save_high_scores()
{
    local a k split_index=0 name dum=() val=() filename="PharFoHighScores.txt"
    ls | grep -q "$filename" || printf "" > $filename
    for k in {1..10}
    do
        a="$k)"
        dum[k]="$(cat $filename | grep $a)"                          # Get line "k) <score> - <player>"
        val=$(printf "%s\n" ${dum[k]#*)} | grep -o -m1 -E '[0-9]+')  # Remove '*)' from string and find first number
        ((split_index==0)) && ((val<=$1)) && split_index=$k
    done
    ((split_index==0)) && return || { printf "What's your name, explorer?\n"; read name; }
    case $name in
        '') name="An unknown traveler..." ;;
    esac
    #sorted_vals=$(printf "%s\n" ${arr[@]} | sort -n) # Sort vals
    printf "Pharaoh's Fortune Top 10\n======================\n" > $filename
    for k in {1..10}
    do
        if ((k<split_index)); then
            printf "$k)${dum[k]#*)}\n" >> $filename
        elif ((k==split_index)); then
            printf "$k) $1 - $name\n" >> $filename
        else
            printf "$k)${dum[$((k-1))]#*)}\n" >> $filename
        fi
    done
    printf "======================\n" >> $filename
    tput cuu1
    printf "Congrats! $name is #$split_index all-time! [see ${filename}]\n"
    cat $filename
}

record_stats()
{
    local filename="PharFoLastAdventure.txt"
    for stat in time steps jewels curse_steps curse_stack
    do
        eval local num='$'$stat
        printf "%4d    " $num >> $filename
    done
    for find in gem trap curse mummy
    do
        eval local found='$'${find}_found
        printf "%4d    " $found >> $filename
```

```
        done
        for item in $item_list
        do
            eval local used='$'${item}_used
            printf "%4d    " $used >> $filename
        done
        printf "\n" >> $filename
    }

    # -----------------
    # -=- Main loop -=-
    # -----------------
    start_game
    load_map
    while ((gameOn))
    do
        draw_map 0
        count_map

        ((!gameOn)) && break

        if ((resetMap)); then
            resetMap=0
            eyeofHorus=0
            load_map
        else
            get_key
            move_dude
        fi

        record_stats

        ((steps += weMovin?1:0))
        ((time++))
    done

    showStuff=0
    tput cup 5 0
    draw_pyramid -1 -1
    printf "\b${black_bg}"
    printf "Thanks for playing!\nFinal score is $current_score\n"
    save_high_scores $current_score
    printf "[n] New game ... [d] Detailed results ... [ENTER] Exit\n"
    local whatNext
    read -n 1 whatNext && printf "\b"
    case $whatNext in
        [Dd]) cat PharFoLastAdventure.txt ;;
        [Nn]) return ;;
    esac
    newgame=0
}

declare -i newgame=1
while ((newgame))
do
    phar_fo
done
#####################################
```