# The game of Hunt the Wumpus

The classic Hunt the Wumpus game ([wiki](https://en.wikipedia.org/wiki/Hunt_the_Wumpus)) evolved from a text-oriented hide-and-seek adventure game to a graphical game on the early microcomputers to this version which is loosely inspired by the classic. This version is a thoroughly updated, modern version

* It uses a touch screen for controls
* The graphics are all high-resolution
* You send a drone quadcopter to take a picture of the animal instead of shooting it with a corked arrow
* The game is a “number of wins out of 10” format with an increasing difficulty. You get more points for having more pictures of different monsters. Wumpus isn’t an actual option because there’s no Unicode Wumpus image.

## Game Play

The goal is to send your Quadcopter to take a picture of the cave’s monster. But the monster is dangerous, so you can’t go into the room with it. Some rooms have pits (avoid those, too!) and some rooms have bats which will randomly move you to another room.

Your exploration is a lesson in logic. You will be told if a bat, pit or Wumpus is nearby (in all cases, ”nearby” means an adjacent room)

You start in a room at random (but it won’t have either a monster or a pit).

## Data structures

A “room” is the most important data structure. Each room is an array, and you pull data out via numerical index (sorry, I wish it used something more fancy!)

1: Room index

2, 3: ROW+COL location (1-based; [1,1] is lower-left)

4, 5, 6, 7: NESW rooms (0==no room)

8: Circle for the room

Rooms are put into the global Map() array. MapNCol and MapNRow are the number of rows and columns.

## Sprints

|  |  |
| --- | --- |
| Time | Work |
| Morning (1 h) = 60m | Basic game setup; basic data structure concept with the “room” and its contents. Create rooms, bats, pits, and monsters. |
| 6:00 🡪 7:18 = 78m | Add function to display a single room (but not the doors)  Add function to display a corridor |
| 8:10 🡪 9:12 = 62m | Updater corridor function to make wider corridors  Add current location  Add ability to display the doors (there are exactly 4 doors at all times; the doors move based on which room you are in)  Add click for doors to move to the correct room + display corridor  Make the blue dot go away! |
| 9:18 🡪 ??  Delay thanks to recursive functions. | Start in a random room  Display the current set of “issues” (e.g. detect bat)  Created BWarn MWarnn PWarn for the bat, monster and pit warning indicators. MakeWarning() sets them up. Added DisplayWarnings(roomIndex) to set when items are near and the utility function RoomHasNear (roomIndex, lookFor, distance). When distance=0, just look in the one room. Distance=1 means look 1 room away, 2 means up to 2 rooms away, etc. This is a recursive function. (Or would be, if recursive functions were a thing in BC BASIC) |
| 8:30 🡪 9:11 41m | The room circles are on top of the doors instead of underneath. Perhaps draw the circles ahead of time with opacity=0 and then change the opacity |
| 9:12 🡪 9:27 16m | The Bat will move you to a new room  Bat=move to new random room  Created **GoToRoom**(roomIndex) to move to a new room |
| 8:35 🡪 9:39 64m | Add a message dialog for the user (that auto-fades)  Created **SetMessage**(msgbox, msg) where msgbox is GLOBAL MsgBox and msg is the text to set. |
| 9:40 🡪 10:06 26m | Let the user take a picture of the monster  Click on a movie camera image and then any circle  Created **State**  s=splash screen   r=running c=setting camera e=end  f=finished (with the entire streak)  Drone: ❊ 🚁 🎥 🦅 📸 |
| 3:40 🡪 4:33 53m | GLOBAL GameResult   0=dead (monster) 1=dead (pit)   10=not dead (ran out of pictures)   20=won (monster picture)  (-1 is the game is still running) GLOBAL NPicturesLeft (starts at 10)  GLOBAL PictureGallery()  Created InitGame (g) function and moved init code into it.  Simple end screen (EndSceneStart/EndSceneEnd)  New e state added |
| 5:32 🡪 6:15 43m  7:57 🡪 9:07 70m = 113m  +more randomly | MsgBox global doesn’t need to be passed into the message functions, and a new message box is always made Discovered that (object=0) is always TRUE! But we can do (0=object) and that works fine.  Display the monster picture gallery and current score FUNCTION InitGameStreak() DIM PictureGallery() FUNCTION **UpdatePictureGallery**()  PictureGalleryText = 0 **NPicturesLeft (=10)**  Score = 0 REM Set the Score +call **UpdateScore**() to update UI ScoreText = 0 **NCavesLeft** = 6 and the CaveText  Set up with a multi-game streak –try to get pictures of them all, do a score, have different numbers of rooms. |
| 7:47 🡪 … | Fix ups for how many games left in the streak, nicer scoreboard, say how well you did in the streak and add more fun to the game. |
| 9:30 🡪 10:00 30m | Improve color scheme  Background is LIGHT\_TAN = “#F7EBD5” 247 235 213  Text VDARK\_BROWN  MessageBox background = DARK\_TAN  Dungeon circles = DARK\_BROWN = “#654320”  Dungeon walls also DARK\_BROWN  Dot color is also DARK\_BROWN |
| 8:00 🡪 9:18 78m | Create start screen with the name, music, etc.  Name is Hunt the Wumpus  Function “SplashScreen(g)” and state “s” (splash) |
| 6:51 🡪 8:50 119m | Have better “end of streak” message  NMonsters (total monsters in streak)  NDoors (number of rooms visited)  TEST\_ArrayIndex  : How many rooms visited per monster? (lower=better)  FUNCTION FinishedScreen(g) state “f” (finished)  Starting a game in a streak should not show the HtW message |
| 6:52 🡪 8:52 120m  7:35 🡪 9:02 87m | Score panel is either a panel or a mini-status line  Insert picture at start of array, not end  Better size control: should be playable on 950 (350w)  This includes changing the size and number of rooms.  Size is S(small) or M(medium) |
| 8:17 🡪 | Set up a “maze” instead of the always grid – remove rooms, include a “via” table. |
|  | Add more fun! Improve the scoring! Pick different monsters! Set the number of pits, etc based on size + random + difficulty! |
| Done! Took about 120m | Gesture control for Continuum |

## Improvements needed for BC BASIC

As part of making the game, a number of failings in the language were discovered.

1. Recursive functions just plain fail (the variables are all cleared). They should either work correctly, or be detected. Silent failure is bad
2. Object=1 (where object is like a Rectangle) returns the object! Which is true!
3. Would be great is there was a way to make some of the graphics fall to the back or pop to the front. For example, the MsgBox is constantly recreated so that it’s guaranteed to be at the front. It would be nice if it could be made early in the process, but floated to always be at the top, or if new items could be made at the back.
4. IF statements where the first statement is one-line and there’s an ELSE are wrong but aren’t shown as wrong (IF () THEN … <CR> ELSE <CR> … END IF will do the ELSE statements always!
5. IF (a AND b AND c) THEN … doesn’t compile