Arduino "Hunt The Wumpus"

Created by Dan Malec

https://learn.adafruit.com/arduino-hunt-the-wumpus

Last updated on 2023-08-29 02:11:51 PM EDT

Table of Contents

Overview & Parts

You can relive the early days of computer gaming on your Arduino with Hunt the Wumpus (). This game is a particularly good fit for the RGB LCD shield:

- Different screens can have different backlight colors
- The LCD is large enough for selecting caves to move to or shoot into
- The Wumpus, bat, and pit can all have custom characters
- The D-Pad is well suited to handling menu navigation

You will need:

- Adafruit RGB LCD Shield (http://adafru.it/716)
- Arduino Uno (http://adafru.it/50)

Of course, you do run the risk of taking a wrong turn and being eaten by a Wumpus!

Code

While coding Hunt the Wumpus, I ran into a problem with one of my functions. I wanted to use an enum () to represent different hazards (bats, pits, Wumpus) and have a common function to check a given cave for hazards and return the enum of the first one found. This helps with showing hazards in neighboring caves as well as checking a cave before the player moves into it.

I started by declaring a function which returns an enum:

```
HazardType check_for_hazards(uint8_t room_idx) {
  if (room_idx == bat1_room || room_idx == bat2_room) {
    return BAT;
  } else if (room_idx == pit1_room || room_idx == pit2_room) {
```

```
    return PIT;
  } else if (room_idx == wumpus_room) {
    return WUMPUS;
  } else {
    return NONE;
  }
}
```

Unfortunately, this resulted in the following error message:

```
Hunt_The_Wumpus:-1: error: 'HazardType' does not name a type
```

It turns out that this is a known issue with a documented workaround () and the fix was as simple as adding a header file Hunt_The_Wumpus.h to my project which contains the enum declaration and a function prototype:

```
enum HazardType { NONE=0, BAT=1, PIT=2, WUMPUS=4 };

HazardType check_for_hazards(uint8_t room_idx);
```

And including the file at the top of the main Hunt_The_Wumpus.ino file:

```
#include "Hunt_The_Wumpus.h"
```

Another interesting area of the code is related to reading button presses. The RGB LCD shield library provides a function for reading which buttons are currently pressed as a bitmask. For menu navigation, it's important to understand clicks. By using a static unit8_t to store the last state of the buttons, it's possible to determine which buttons have been pressed and then released.

```
void read_button_clicks() {
  static uint8_t last_buttons = 0;

  uint8_t buttons = lcd.readButtons();
  clicked_buttons = (last_buttons ^ buttons) &amp; (~buttons);
  last_buttons = buttons;
}
```

Downloads

Download the latest code on Github ()

Here's a 'diff' for monochrome displays!

```
*** Hunt_The_Wumpus.ino.orig 2012-11-24 12:43:10.936042927 -0500
--- Hunt_The_Wumpus.ino 2012-11-24 13:14:11.908126852 -0500
***************
*** 42,47 ****
--- 42,48 ----
#include
```

```
#include "Hunt_The_Wumpus.h"

+ #define MONO 1 // cheapskate with no RGB (or they were out of stock)

//! Map of the cavern layout.
/*!
***************
*** 270,276 ****
--- 271,281 ----
//! Initial game state, draw the splash screen.
void begin_splash_screen() {
lcd.clear();
+ #ifndef MONO
lcd.setBacklight(TEAL);
+ #else
+ lcd.setBacklight(RED); // MONO backlight ON is on RED line
+ #endif
lcd.print(F("HUNT THE WUMPUS"));

state = animate_splash_screen;
***************
*** 357,363 ****
--- 362,370 ----

void begin_bat_move() {
lcd.clear();
+ #ifndef MONO
lcd.setBacklight(BLUE);
+ #endif
lcd.write(BAT_ICON_IDX);
lcd.setCursor(5, 0);
lcd.print(F("Bats!"));
***************
*** 429,439 ****
--- 436,448 ----
lcd.write(ARROW_ICON_IDX);
lcd.print(arrow_count);

+ #ifndef MONO
if (adjacent_hazards) {
lcd.setBacklight(YELLOW);
} else {
lcd.setBacklight(TEAL);
}
+ #endif

lcd.setCursor(1, 1);
for (int i=0; i--&gt;"));

arrow_count--;
***************
*** 575,581 ****
--- 590,598 ----

void draw_game_over_screen(uint8_t backlight, __FlashStringHelper *message, uint8_t
icon) {
lcd.clear();
+ #ifndef MONO
lcd.setBacklight(backlight);
+ #endif
lcd.print(message);
lcd.setCursor(0, 1);
lcd.write(icon);
```