Die analoge Uhr

Description

Es kann vorkommen, dass man im Spiel eine analoge Uhr anzeigen möchte, die entweder eine vorgegebene oder die aktuelle Uhrzeit anzeigt. Einige Elemente der Uhr, wie etwa den Hintergrund oder die Zeiger, kann man mit Sprites realisieren. Technisch eleganter ist es aber, wenn man die Zeichenfunktionen von GameMaker benutzt.

Das Ziel dieses kleinen Tutorials ist diese analoge Uhr.

Vorbereitung

Starte ein **neues Projekt** und erstelle **einen Font** mit dem Namen *fnt_zeit*. Anschließend erstellst Du **einen Raum** und **ein Objekt** mit dem Namen *obj_uhr*, welches Du in den Raum legst. Mein Raum hat eine Größe von *420×420* Pixel. Euer kann gerne größer sein. Damit das Beispiel funktioniert, solltest Du keinen kleineren Raum wählen oder den Code ggf. anpassen.

Ab jetzt arbeiten wir nur noch im Objekt *obj_uhr* weiter.

Der Code

Du brauchst drei Events. **Create**, **Draw** und **release <Escape>**. Letzteres brauchst Du nur, um die Uhr bequem zu schließen. Ziehe per D&D in alle drei Events die **Execute-Code Action** hinein.

Event release <Escape>

Hier landet lediglich der Code, um das Programm zu beenden:

```
game_end();
```

Event Create

Nun definieren wir die Ausgangsvariablen.

```
// Wir positionieren die Uhr in der Mitte des Raumes
xx = room_width / 2;
yy = room_height / 2;

s_laeng = 170;      // Länge des Sekundenzeigers
s_breit = 3;        // Breite des Sekundenzeigers

m_laeng = 140;      // Länge des Minutenzeigers
m_breit = 5;        // Breite des Minutenzeigers

h_laeng = 100;      // Länge des Stundenzeigers
h_breit = 8;        // Breite des Stundenzeigers

aussen = 200;       // Außendurchmesser
innen = 180;        // Innendurchmesser
```

Event Draw

Hier wird die eigentliche Uhrzeit ermittelt und die ganze Uhr, ohne die Verwendung von Sprites, angezeigt. Zeiger und Beschriftung werden in Echtzeit gezeichnet. Bitte beachte, dass Du den Font erstellt und entsprechend bezeichnet hast.

```
// Zifferblatt-Rand
draw_set_color(c_dkgray);
draw_circle(xx, yy, aussen, false);

// Zifferblatt-Innen
draw_set_color(c_gray);
draw_circle(xx, yy, innen, false);

// Zifferblatt Punkt hinten
draw_set_color(c_black);
draw_circle(xx, yy, 20, false);
```

```
// Sekunden
seconds_dir = 90 - current_second * 6;

draw_set_color(c_black);
draw_line_width(xx, yy, xx + lengthdir_x(s_laeng, seconds_dir), yy + lengthdir

// Minuten
minutes_dir = 90 - current_minute * 6;

draw_set_color(c_dkgray);
draw_line_width(xx, yy, xx + lengthdir_x(m_laeng, minutes_dir), yy + lengthdir

// Stunden
hour_dir = 90 - current_hour * 30 - current_minute / 10 * 5;

draw_set_color(c_red);
draw_line_width(xx, yy, xx + lengthdir_x(h_laeng, hour_dir), yy + lengthdir_y(

// Zifferblatt Punkt vorne
draw_set_color(c_dkgray);
draw_circle(210, 210, 12, false);

// Zifferblatt Linien und Zahlen
draw_set_font(fnt_zeit);
draw_set_valign(fa_middle);
draw_set_halign(fa_center);

a = 3; // Uhrzeit auf dem Ziffernblatt. Wir beginnen bei drei und zählen gegen

draw_set_color(c_black);
for (i = 0; i < 360; i += 30)
{
    draw_line_width(xx + lengthdir_x(innen - 15, i), yy + lengthdir_y(innen -
    draw_text(xx + lengthdir_x(innen - 32, i), yy + lengthdir_y(innen - 32, i)

    // Bei 1 sollte die Zählung auf 12 springen
    if (a = 1)
    {
        a = 12;
    } else {
        a--;
    }
}

draw_set_color(c_dkgray);
for (i = 0; i < 360; i += 6)
{
    draw_line_width(xx + lengthdir_x(innen - 10, i), yy + lengthdir_y(innen -
}
```

Auch wenn der Code kommentiert ist, sollte noch ein bisschen Erklärung folgen. Mit den Befehlen, *current_second*, *current_minute* und *current_hour* wird die aktuelle Zeit ermittelt. Die entsprechenden Variablen sind mit einem _dir am Ende betitelt. Das steht für Direction, also die Richtung, in welche der Zeiger zeigt. 0° zeigt beim GameMaker immer nach rechts, weshalb wir die Richtung mit 90 beginnen,

was nach oben zeigt.

Merke: GameMaker berechnet die Richtung immer gegen den Uhrzeigersinn, beginnend bei 0° für rechts, 90° für oben, 180° für links und 270° für unten.

Gezeichnet werden die Zeiger mit *draw_line_width*. Das ist eine Erweiterung von *draw_line* und ermöglicht es, der Linie eine Breite zu geben. Der Startpunkt der Linien ist immer *xx* und *yy* aus dem **Create-Event**, in unserem Beispiel ist es die Raummitte. Die Schwierigkeit liegt nun darin, die korrekte Endposition der Linie zu bestimmen. Dabei helfen die Befehle *lengthdir_x* und *lengthdir_y*. Diese Befehle geben die horizontale (x) und vertikale (y) Komponente des Vektors durch die angegebene Länge an. Wir müssen also nur noch die Länge der Linie, die wir im **Create-Event** definiert haben, und den Winkel, den wir im **Draw-Event** berechnet haben, angeben. Für die Sekunde ist es die Zeile 17, welche die Linie Zeichnet. Hier sei sie noch einmal hervorgehoben:

```
draw_line_width(xx, yy, xx + lengthdir_x(s_laeng, seconds_dir), yy + lengthdir
```

Wenn man die Bedeutung der Zeile aussprechen würde, würde es wie folgt lauten: „Zeichne eine Linie, die bei *xx* und *yy* beginnt. Der Endpunkt ist auf der x-Koordinate *xx* plus die Richtung Länge des Sekundenzeigers und der errechneten Richtung *seconds_dir*. Der Endpunkt ist auf der y-Koordinate *yy* plus die Richtung Länge des Sekundenzeigers und der errechneten Richtung *seconds_dir*. Die Linie hat eine Breite von *s_breit*.”

Übrigens: Nach der gleichen Methode werden auch die Zeiten und Linien auf dem Ziffernblatt gezeichnet. Dafür verwenden wir dann eine **for-Schleife**.

Analoge Uhr

1 Datei(en) 306.34 KB
Download
Date Created
18. September 2016
Author
sven