Buttons für Jedermann

Description

Es gibt verschiedene Möglichkeiten, tolle Buttons in GameMaker zu erzeugen. Die nachfolgende Methode funktioniert nicht nur in Studio, sondern auch in älteren Versionen.

Fast jedes Spiel braucht Buttons, mit denen der Spieler bestimmte Aktionen wählen kann. Egal ob Auswahl von Missionen, der Gang ins Optionsmenü oder das Verlassen des Spiels: Mit Buttons sieht das meist besser aus, als mit irgendwelchen Tastenkombinationen.

Im Prinzip kann man es sich dabei sehr einfach machen: Wir nehmen eine Grafik, importieren sie in ein leeres Objekt und sagen dem Objekt nur, was es tun oder auslösen soll, sobald die linke Maustaste gedrückt oder losgelassen wurde. Doch das ergibt noch keinen Button.

Wunschliste

Ein Button in einem Spiel sollte mindestens so gut funktionieren, wie im Betriebssystem oder auf einer Webseite. Das ist der Spieler gewohnt und mit den Eigenschaften kann er auch etwas anfangen. Um dies realisieren zu können, braucht unser Button drei Zustände:

1. Default-Zustand. Das heißt, der Zustand, wenn noch nichts passiert. Der Button existiert nur, kein Mauszeiger hat ihn je berührt.
2. Mouse-Over-Zustand. Wenn der Mauszeiger drüber geht, muss der Button reagieren und sich zumindest grafisch ändern. In Spielen ist es immer noch ganz nett, wenn sich auch akustisch etwas tut, aber die visuelle Änderung ist Pflicht, weil der Spieler damit das Signal bekommt, dass er mit dem Button etwas machen kann.
3. Mouse-Click-Zustand. Wenn die Maus geklickt wird, muss ebenfalls eine optischer Änderung des Buttons erfolgen. Wenn ein akustisches Signal vorgesehen ist, dann würde ich das erst auslösen, wenn die Maustaste losgelassen wird. Das ist zwar nicht Bestandteil des Tutorials, unser Button lässt sich aber am Ende sehr einfach modifizieren.
4. Egal, ob der Mauszeiger klickt oder ohne Klick den Button verlässt, muss der Button wieder in ihren ursprünglichen Zustand versetzt werden.
5. Der Mauszeiger muss sich ebenfalls ändern. Sobald der Zeiger auf dem Button ist, muss sich die

Grafik oder zumindest ihre Farbe ändern. Wenn der Mauszeiger den Button verlässt, muss der Mauszeiger in den ursprünglichen Zustand zurückkehren.

Vorbereitung Grafiken

Unser Ziel ist ein Button für ein Hauptmenü, in dem man Spiel, Optionen, Beenden und vergleichbares auswählen kann. Das Prinzip, welches wir anwenden, lässt sich auf jede andere Situation übertragen. Du kannst nach dem Tutorial mit diesem Wissen auch Buttons für bestimmte Spielsituationen erschaffen.

Wichtig ist uns, dass die Buttons möglichst flexibel sind, vor allem im Menü. Wir wollen keine Buttons, deren Grafiken beschriftet sind, weil wir dann für jedes Button und für jede Sprache, falls es ein mehrsprachiges Spiel wird, neue Grafiken brauchen. Die Beschriftung kommt also durch den Code, nicht durch die Grafik.

spr_button

In diesen Sprite kommen die drei Grafiken rein für Default, Mouse-Over und Klick. Die Grafik zeigt die drei Zustandsänderungen und wird, wie eben erwähnt, nicht beschriftet. Die Grafik in der angehängten Beispieldatei hat eine Auflösung von 260×48 Pixel. Du kannst deine Grafik gerne kleiner machen, aber nicht zu klein, schließlich soll ja auch eine Beschriftung rein.

Die Ausrichtung (**Origin**) sollte auf Center stehen und die Maske die ganze Grafik beinhalten.

spr_mousecourser01

Dies ist unser Mauszeiger im Default-Zustand. Mein Mauszeiger hat eine Auflösung von 22×32 Pixel, Du kannst aber gerne einen eigenen verwenden. Auch hier sollte die Maske das ganze Bild umschließen, die Ausrichtung sollte aber an der Mausspitze sein. In meinem Fall ist das bei 0 / 0.

spr_mousecourser02

Dies ist nun die Grafik, die angezeigt wird, sobald der Mauszeiger einen Button berührt. In meinem Fall ändert sich nur die Farbe, Du kannst aber auch aus dem Zeiger eine Hand machen oder dir etwas anderes einfallen lassen. Die Eigenschaften der Grafik sind wie beim Ausgangszustand.

Sonstige Vorbereitungen

fnt_menue

Wir möchten die Buttons beschriften, brauchen also eine Schrift. Für das Beispiel benutze ich *Arial* in der Größe 12. Du kannst aber gerne eine kreativere Schrift verwenden. Wichtig ist nur, dass man sie gut lesen kann.

obj_button_parent

In diesem Objekt werden Dinge definiert, die für alle Menübuttons Gültigkeit haben werden. Wer sich mit der Vererbung von Objekteigenschaften noch nicht auskennt, sollte sich keine Sorgen machen. An dieser Stelle reicht es nur, nachzubauen. Das Thema der Vererbung wird in einem anderen Tutorial ausführlicher behandelt. Wenn Du das Objekt erstellt hast, lade als Sprite *spr_button* ein. Dann kannst Du das Objekt vorläufig schließen.

obj_button1 und obj_button2

Ja genau, wir machen zwei Buttons. Setze auch hier als Sprite *spr_button* ein. Wichtig ist, dass Du unter **Parent** das Objekt *obj_button_parent* auswählst.

obj_mouse

Unser Mauszeiger bekommt ein eigenes Objekt. Als Sprite wählst Du *spr_mousecourser01* aus und als **Depth** trägst Du *-100* ein. Damit garantierst Du, dass der Mauszeiger immer über den Buttons ist.

Wenn Du die Maus in mehreren Räumen verwenden möchtest, dann mach noch einen Haken bei **Persistent**.

room_buttons

Wie immer, gibt es auch hier einen Testraum. Meiner hat eine Auflösung von 600×200 Pixel und einen schwarzen Hintergrund. Hier positionierst Du die Maus und die Objekte *obj_button1* und *obj_button2*.

Hier kommt die Maus

In der Maus brauchen wir drei Events.

Event Create

```
window_set_cursor(cr_none); // Wir schalten den Windows-Mauszeiger aus
mouseOff = true;            // Die Maus ist nicht auf einem Button
visible = true;             // Die Maus ist sichtbar
```

In der ersten Zeile machen wir den Windows-Mauszeiger unsichtbar. Dann definieren wir eine Variable die uns sagt, ob die Maus aus (also nicht auf einem Button) oder an ist. Zuletzt stellen wir zur Sicherheit noch die Sichtbarkeit ein. Der Befehl ist interessant, wenn man die Maus in manchen Räumen abschalten will. Dann muss man ein persistentes Objekt nicht zerstören. Im Fall des Mauszeigers reicht es, das Objekt unsichtbar zu machen.

Event Step

```
x = min(max(mouse_x, 0), room_width);
y = min(max(mouse_y, 0), room_height);
```

Mit diesen beiden Zeilen wird die Maus von der Grafik verfolgt. Der Befehl **min** liefert den kleinsten Wert von mehreren möglichen Werten zurück. In diesem Fall ist es die entsprechende

Mauskoordinate, also x bzw. y, und die Raumbreite bzw. Raumhöhe. Wenn *mouse_x* kleiner ist als die Raumbreite, ist x gleich der Koordinate *mouse_x*. Wenn nicht, wird die Maus automatisch dort positioniert, wo der Raum endet. Gleiches gilt für die *mouse_y*-Koordinate.

Event Draw

```
if (mouseOff)
{
    draw_sprite(spr_mousecourser01, -1, x, y)
} else {
    draw_sprite(spr_mousecourser02, -1, x, y)
}
```

Hier bekommt die Maus ihren richtigen Sprite. Ob sie auf einem Button liegt oder nicht, erfährt sie vom Button selbst.

Das Eltern-Button

Im GameMaker ist es möglich, Objekteigenschaften zu vererben. Das ist eine super Sache. Wenn wir viele, sehr ähnliche, aber nicht identische Objekte haben, können wir die Eigenschaften, die identisch sind, vererben und müssen diese nicht in jedem Objekt neu erstellen. Bei der Erstellung von Objekten ist der eingesparte Aufwand oft nicht einmal so hoch, aber wir merken es, wenn wir Eigenschaften ändern möchten. Wir müssen nur in das Elternobjekt gehen, die Änderung vornehmen und sie wirkt sich auf alle Kinder aus. Im Elternobjekt kann man auch sehen, welche Objekte, also welche Kinder, dazu gehören.

Beschriftungen und andere Eigenschaften sind bei den Buttons unterschiedlich, aber es gibt Eigenschaften, die identisch sind. Wichtig ist zu wissen, dass die Vererbung auf Events basiert. Wir vererben den Kindern nur die Events, die sie selbst nicht haben. Gibt es Beispielsweise im Eltern-Objekt ein Draw-Event und bei den Kindern ebenfalls eines, wird das Draw-Event des Eltern-Objekts nicht vererbt sondern von den Kindern überschrieben.

In unserem Fall gibt es drei Event, die alle gleich haben. Bei den Kindern wird es drei weitere Events geben, die sich unterscheiden. Auch wenn in den Objekten nur jeweils drei Events angezeigt werden, verhält sich der Button im Spiel so, als hätte er sechs Events, nämlich die drei Eigenen und die drei vererbten.

Um genug Übersicht zu haben, setzen wir das Eltern-Objekt nicht im Spiel ein. Es dient lediglich der Vererbung.

Event Create

```
mouseOver           = true;  // MouseOver ist an
obj_mouse.mouseOff  = false; // Wir sagen der Maus, dass sie auf uns drauf ist
```

Das Verhalten, wenn die Maus den Button berührt, ist auch immer gleich. Wir setzen die eigene Variable *mouseOver* auf true, um dem **Draw-Event** zu sagen, dass eine andere Grafik angezeigt werden muss. Der Maus sagen wir ebenfalls, dass sie auf dem Button ist, damit sich auch hier die Grafik ändert.

Event Mouse Leave

```
mouseOver           = false;
mouseKlick          = false;
obj_mouse.mouseOff  = true;
```

Wenn die Maus nicht mehr auf dem Button ist, wird der Ursprungszustand für Button und Maus hergestellt, also die Variablen auf den Ausgangswert gesetzt.

Übrigens kann man an diesen Stellen, wenn man will, auch einen Soundeffekt abspielen. Im **Mouse Enter Event** ist das besonders schick.

Die Kind-Buttons

Ich zeige hier nur die Funktionen von *obj_button1*. Für *obj_button2* sind sie nahezu identisch, aber was beim Klick passieren soll und welcher Text angezeigt wird, kannst Du selbst bestimmen.

Event Left Button

```
mouseOver   = true;
mouseKlick  = true;

// Hier kommt dann, was der Button machen soll
// Man könnte zum Beispiel einen Soundeffekt abspielen
```

Wenn mit der linken Maustaste geklickt wird, werden erst einmal die Variablen gesetzt, damit die Grafik richtig angezeigt werden kann. Anschließend, wo der Kommentar steht, wird angegeben, was nach dem Klick passiert. Hiermit sind Sounds oder optische Effekte gemeint. Was tatsächlich mit dem Mauszeiger ausgelöst wird, erscheint im nächsten Event.

Event Left Released

```
mouseKlick = false;

// Hier kommt dann, was beim loslassen passieren soll.
```

Hier kann man das Spiel beenden, den Raum verlassen oder andere Aktionen auswählen.

Event Draw

```
if (!mouseOver)
{
    draw_sprite(spr_button, 0, x, y)
} else {
```

```
    draw_sprite(spr_button, 1, x, y)
}

if (mouseKlick)
{
    draw_sprite(spr_button, 2, x, y)
}

// Hier kommt die Beschriftung hin
draw_set_font(fnt_menue);           // Schrift
draw_set_color(2532356);            // Farbe
draw_set_halign(fa_center);         // Zentriert
draw_set_valign(fa_middle);         // Zentriert

if (mouseKlick = true)
{
    draw_text(x, y, „Button 1 klick");   // Beschriftung
} else if  (mouseOver){
    draw_text(x, y, „Button 1 MoseOver");   // Beschriftung
} else {
    draw_text(x, y, „Button 1");   // Beschriftung
}
```

Im oberen Bereich wird lediglich das richtige Sprite angezeigt, je nach Zustand der Variablen. Darunter kommt die Beschriftung. Wir setzen Schrift, Farbe aus Ausrichtung. Dann, je nach Variable, können wir sogar den Text ändern. In den meisten Fällen ist das nicht sehr praktikabel, den Text zu ändern, aber stattdessen könnte man hier auch eine andere Schrift oder Farbe verwenden. Der Kreativität sind hier kaum Grenzen gesetzt.

Fertig! Nun kannst Du das Programm starten und Dich an deinen Buttons erfreuen. Das war zwar ein gewisser Aufwand, aber dafür hast Du nun Buttons, die Du immer wieder verwenden kannst. Sie lassen sich super anpassen und sehr leicht zu handhaben.

Buttons

1 Datei(en) 462.25 KB
Download
Date Created
2. November 2016
Author
sven