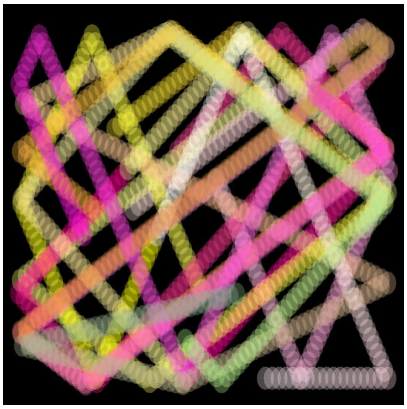
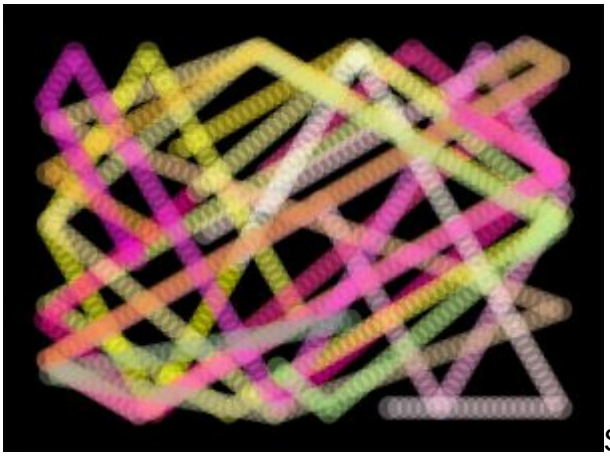


Shadebobs

Description

Mit dem GameMaker kann man allerlei Effekte programmieren. Sogar Effekte aus den 1980er und 1990er Jahren kann man authentisch imitieren. Heute nehmen wir uns die Shadebobs vor.



Shadebobs sind im Prinzip Sprites, die sich schnell auf dem Bildschirm bewegen und dabei von ihrer ursprünglichen Position nicht gelöscht, sondern kopiert werden. Wenn mehrere solcher Sprites über den Bildschirm huschen und vom Bildschirmrand abprallen, ergibt das einen ziemlich coolen Effekt.

Wozu brauche ich Shadebobs?

Erst einmal ist es ein ziemlich simpler Effekt, der sich sehr gut dafür eignet, in die Effektprogrammierung einzusteigen. Dieses Tutorial ist auch extra für Anfänger aufgebaut. Die Methode, die hier gezeigt wird, ist vielleicht nicht die performanteste (das ist sie ganz sicher nicht), aber für Anfänger am besten geeignet.

Der zweite Grund ist ganz einfach: Weil es cool aussieht! Ne, ganz im Ernst. Solche Effekte eignen sich toll für Retro-Spiele. Man kann mit verschiedenen alten Effekten etwas bauen, das nach einem

Cracktro aussieht und dies vor dem Hauptmenü des Spiels zeigen. Man kann solche Effekte auch zeigen, wenn man die Credits bringt, oder generell einen Abspann. Es gibt genug Möglichkeiten, coole, alte Effekte in einem Spiel unter zu bringen, wenn man das nur will.

Vorbereitung

Erst einmal brauchst Du einen **Raum**. Ich verwende eine **Raumgröße** von 1024×768. Der **Hintergrund** ist *Schwarz*, **Speed** sollte man immer auf *60* stellen.

Als Sprite brauchst Du ein kleines Kollisionsobjekt. Ein Kreis mit 20×20 Pixel ist für unser Beispiel genau richtig. Stelle die **Origin** auf *Center*, die **Maske** auf *Automatic* und bei **Shape** *Elipse*. Das Sprite nennst Du dann *spr_collision_shadebob*.

Nun brauchen wir noch einen zweiten Sprite. Dabei handelt es sich um eine später unsichtbare Wand mit einer Größe von 64×64 Pixel. Erstelle eine entsprechend große Fläche und fülle sie mit einer Farbe deiner Wahl. Es sollte sich lediglich vom Hintergrund unterscheiden. Die Wand wird man später ohnehin nicht sehen. **Origin** kannst Du bei *0* lassen, die **Maske** ist auch in Ordnung. Das Objekt nennst Du *spr_collision_wall*.

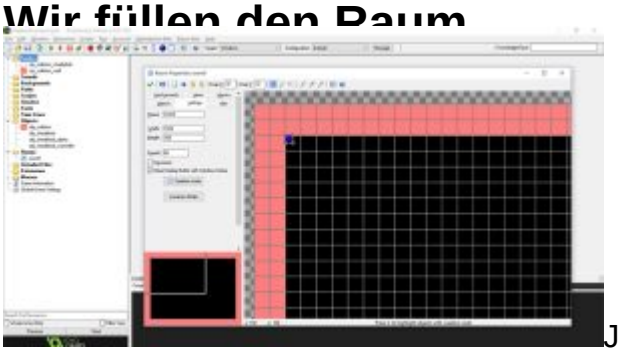
Jetzt brauchen wir noch vier Objekte. Das erste Objekt nennst Du *obj_collision*. Als Sprite wählst Du *spr_collision_wall* aus und kreuzt **Solid** an. Bei **Visible** entfernst Du den Hacken. Wir wollen ja eine unsichtbare Wand.

Das zweite Objekt nennst Du *obj_shadebob* und das dritte *obj_shadebob_alpha*. *obj_shadebob* zeichnet nur die Grafik und verschwindet mit der Zeit wieder. *obj_shadebob_**alpha*** saust über den Bildschirm und setzt die Sprites. Dem Objekt *obj_shadebob* geben wir noch die Maske des entsprechenden Sprites, also *spr_collision_shadebob*.

Noch kurz etwas zur Namensgebung. Das **alpha** hat zwei Bedeutungen. Erstens steht es dafür, dass dieses Objekt der Leader ist, zweitens, dass es unsichtbar ist. Ich hätte auch unsichtbarer_leader nehmen können.

Das letzte Objekt nennen wir *obj_shadebob_controller*. Damit steuern wir die Instanzen des Objektes *obj_shadebob_alpha* und das Objekt *obj_shadebob_alpha* erzeugt *obj_shadebob*. Das klingt zunächst verwirrend, ist aber am Ende total einfach.

Wir füllen den Raum



Jetzt gehst Du in den von Dir erstellten Raum. Mit dem Objekt *obj_collision* erzeugst Du einen Rahmen, wie Du es auf dem Screenshot sehen kannst. Dann

setzt Du den Controller *obj_shadebob_controller* in die linke obere Ecke des Rahmens. Nun kannst Du den Raum schließen, wir sind damit fertig.

Events obj_shadebob

Wie oben bereits beschrieben, ist *obj_shadebob* nur der Shadebob selbst. Der macht nichts, außer auf dem Bildschirm zu sein und sich nach einer gewissen Zeit (im Tutorial sind es drei Sekunden) zu zerstören. Dafür brauchen wir drei Events.

Event Create

```
alpha = 0.33;

col = make_color_rgb(80, 80, 80);

alarm[0] = 3 * room_speed;
```

Wir definieren einen Alpha-Wert von 0.33 und die Farbe col. Die besteht aus einem RGB-Wert und für zu einem grauen Ergebnis. Die finale Farbe definieren wir ohnehin wo anders, wir brauchen hier lediglich die Variable. Dann rufen wir noch einen Alarm aus, der nach drei Sekunden ausgeführt wird.

Event Alarm 0

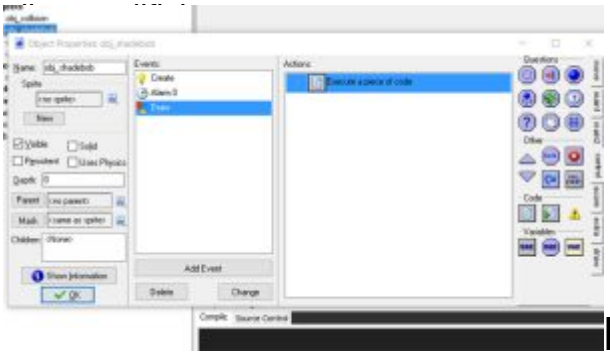
```
instance_destroy();
```

Das war es auch schon. Nach drei Sekunden zerstört sich das Objekt wieder.

Event Draw

```
draw_set_alpha(alpha);
draw_circle_colour(x, y, 20, col, col, 0)
```

Wir setzen der Alpha-Wert aus dem Create-Event und zeichnen nun einen Kreis. Der Radius beträgt 20 Pixel. Beide Farben sind gleich, weil wir hier keinen Verlauf haben wollen. Das lässt sich aber noch



Events obj_shadebob_alpha

Das unsichtbare Objekt *obj_shadebob_alpha* saust später über den Bildschirm und lässt die Shadebobs fallen. Es prallt von den Wänden ab und damit es das machen kann, prüfe noch einmal, ob

unter Mask wirklich *spr_collision_shadebob* eingestellt ist. Nun geht es an die Events, von denen wir ebenfalls drei haben.

Event Create

```
colr = random(255) - 35;
colg = random(255) - 35;
colb = random(255) - 35;
col_forward = true;

dir = random(45) + 15;

direction += dir;

speed = 16;

alarm[0] = 9 * room_speed;
```

Die Variablen *colr*, *colg* und *colb* sind die einzelnen RGB Werte der Farbe, welche wir an den Shadebob weiter reichen werden. *col_forward* ist wichtig für das Zählen der Farbe. Das machen wir dann im Step-Event. dir ist die Richtung, in welche unser Objekt fliegen wird. Auch hier verwenden wir einen Zufallswert. Du kannst hier gerne mit den Zahlen spielen, wenn Du möchtest. Anschließend geben wir diese Richtgung weiter und definieren eine Geschwindigkeit von 16. Der Alarm von 9 Sekunden ist nur dafür da, dass sich der Effekt zerstört. Du kannst ihn beliebig lange anzeigen lassen, wenn Du das willst.

Event Alarm 0

```
instance_destroy();
```

Auch hier zerstören wir nur das Objekt, bzw. die Instanz.

Event Step

```
// Wir generieren eine Farbe. Da wir einen Farbverlauf
// möchten, zählen wir R, G, und B immer einen wert hoch
// bis wir 255 erreicht haben. Dann gehen wir die Farbskala
// zurück, bis wir 1 haben.

if (col_forward)
{
    if (colr <= 255)
    {
        colr += 1;
    }

    if (colr >= 255) && (colg <= 255)
    {
        colg += 1;
    }

    if (colg >= 255) && (colb <= 255)
```

```
    {
        colb += 1;
    }
    if (colb >= 255)
    {
        col_forward = false;
    }
} else {
    if (colr >= 1)
    {
        colr -= 1;
    }

    if (colr <= 1) && (colg >= 1)
    {
        colg -= 1;
    }

    if (colg <= 1) && (colb >= 1)
    {
        colb -= 1;
    }

    if (colb <= 1)
    {
        col_forward = true;
    }
}

col = make_color_rgb(colr, colg, colb); // Wir setzen die Variable aus den Ein

aaa = instance_create(x, y, obj_shadebob); // Nun erzeugen wir den Shadebob...
aaa.col = col; // ...und geben ihm die Farbe als Variable mit

move_bounce_solid(1); // Kollision mit der Wand
```

Der Code ist etwas umfangreicher, weswegen ich ihn kommentiert habe. Vereinfacht gesagt, erzeugen wir mit jedem Step eine neue Farbe, den RGB-Wert schreiben wir in die Variable *col* und dann erzeugen wir eine Instanz von *obj_shadebob* an der Stelle, an welcher gerade die Instanz *obj_shadebob_alpha* ist. Die Variable *aaa* brauchen wir, damit wir an die Instanz die Farbe weiter geben können. Die letzte Zeile im Code besagt nur, dass der Shadebob an Solid-Objekten abprallen soll.

Events obj_shadebob_controller

Jetzt sind wir schon fast fertig, wir brauchen nur noch den Controller um die Instanzen von *obj_shadebob_alpha* zu erzeugen. Außerdem bauen wir noch ein, dass wir das Fenster mit Escape schließen können.

Event Create

```
instance_create(99, 100, obj_shadebob_alpha);
```

```
instance_create(112, 112, obj_shadebob_alpha);
instance_create(700, 168, obj_shadebob_alpha);
instance_create(128, 245, obj_shadebob_alpha);
instance_create(302, 200, obj_shadebob_alpha);
instance_create(121, 300, obj_shadebob_alpha);
instance_create(245, 600, obj_shadebob_alpha);
instance_create(904, 400, obj_shadebob_alpha);
```

Wir erzeugen jetzt einfach Mal 8 Objekte auf dem Bildschirm und verteilen sie ein wenig im Raum. Achte bei den Koordinaten darauf, dass sie nicht auf den Rahmen treffen. Wir haben einen Raum von 1024×768 und einen Rahmen mit einer Dicke von 64. Unsere Maske hat einen Radius von 10 Pixel, was zu einem Mindestabstand von 74 Pixel führt. Auf der Breite dürfen wir also nur Zahlen zwischen 75 und 949 nehmen (*x*) und in der Höhe 75 und 693. Innerhalb dieses Bereiches kannst Du mit den Zahlen spielen. Du kannst sie auch zufällig erzeugen und das sogar in eine Schleife setzen. Diese Optimierung darfst Du gerne als Hausaufgabe sehen.

Event release <Escape>

```
game_end();
```

Das war es auch schon. Nun kannst Du das Projekt starten und sehen, was wir gemeinsam angestellt haben.

Ich fasse noch einmal kurz zusammen, was da eigentlich passiert ist. Das Objekt *obj_shadebob_controller* erzeugt im Raum 8 Instanzen des Objektes *obj_shadebob_alpha.* Diese bewegen sich mit einer Geschwindigkeit von 16 in eine zufällige Richtung durch den Raum. Sie ermitteln eine Farbe, erzeugen in jedem Step die Instanz des Objektes *obj_shadebob* und sagen ihr, welche Farbe sie hat. Wenn *obj_shadebob_alpha* an eine Wand kommt, prallt es einfach ab. Nach 9 Sekunden zerstört sich die Instanz selbst. *obj_shadebob* zerstört sich ebenfalls, allerdings schon nach 3 Sekunden. Bis dahin verharrt es auf seiner Position und versucht möglichst gut auszusehen.

Natürlich lassen sich Bewegungen, Farben und Formen weiter optimieren und variieren. Dabei wünsche ich Dir viel Spaß!

Shadebobs

1 Datei(en) 197.38 KB
Download
Date Created
26. September 2016
Author
sven