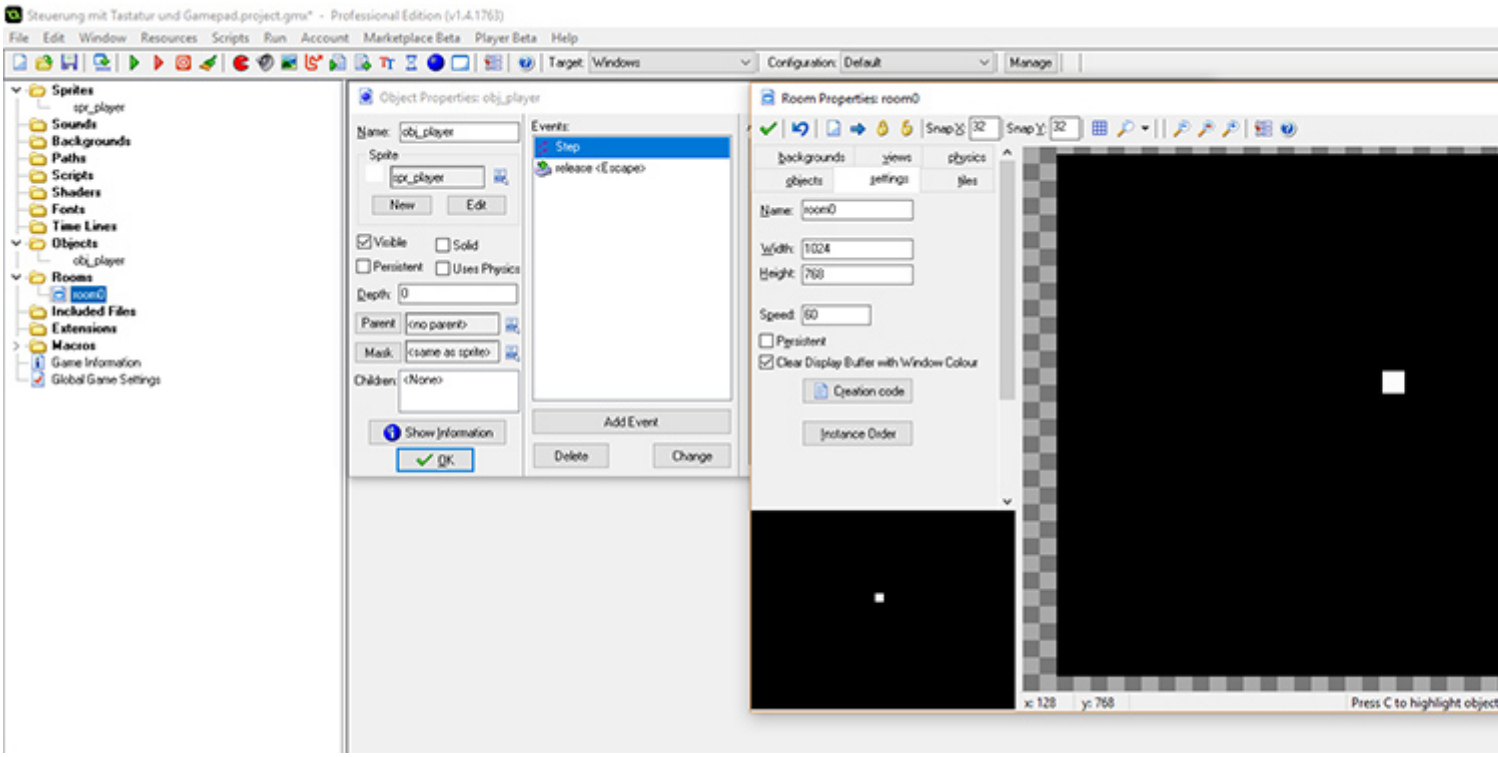
Steuerung mit Tastatur und Gamepad

Description

Wir brauen auf dem gestrigen Tutorial, bei dem es um die Bewegung von Objekten ging, auf, und steuern ein Objekt mit der Tastatur und dem Gamepad.

Bevor wir mit dem eigentlichen Projekt beginnen, solltest Du das gestrige Tutorial nachbauen. Auf dem folgenden Screenshot siehst Du, was Du alles brauchst. Den Code im **Step-Event** kannst Du weglassen.



Du brauchst also ein Sprite mit 32×32 Pixel, ein Objekt und einen Raum. Das Objekt bekommt den Sprite, der Raum einen Speed von 60 und das Objekt wird mittig im Raum platziert. Nun brauchst Du noch die beiden Events. **Step** für die Steuerung des Objekts und **release <Escape>** um wieder aus dem Spiel zu kommen.

Ersteinmal möchten wir das Objekt mit der Tastatur, genauer gesagt mit den Pfeiltasten steuern. Wie wir mittlerweile wissen, gibt es für die Steuerung des Objekts drei Möglichkeiten. Die Steuerung soll so ablaufen, dass die Bewegung erst ausgeführt wird, wenn wir die Taste drücken und sofort aufhört, sobald wir die entsprechende Taste loslassen. Das gilt sowohl für die Tastatur, wie auch für das Gamepad.

Die Tastatur

Man kann auch mit dem Gamepad beginnen, aber in den meisten Fällen wird man auch die Tastatur verwenden und deshalb fange ich gerne damit an. Das Gamepad baut dann auf den Code auf.

Die Ausgangslage ist klar. Wir wollen, dass sich das Objekt bei gedrückter Taste in die entsprechende Richtung bewegt. Wenn wir nach oben drücken, wird der y-Wert geringer, drücken wir nach unten, wird er größer. Gleiches gilt für links und rechts. Nach rechts wird x größer, nach links kleiner.

Wir brauchen somit eine if-Anweisung, welche die Tastatur abfragt und sagt, was passieren soll, wenn die Taste gedrückt wurde. Um die Tastatur abzufragen, gibt es in GameMaker mehrere Befehle. Die drei wichtigsten sind:

- keyboard_check
- keyboard_check_pressed
- keyboard_check_released

keyboard_check_pressed fragt ab, ob die taste gedrückt wird. Sobald man die entsprechende Taste drückt, bekommt der GameMaker das Signal, dass die Taste gedrückt wurde und führt den nachfolgenden Befehl aus. Um ihm wieder auszuführen, muss man erneut drücken. Für die Bewegung in eine entsprechende Richtung ist das ungeeignet, weil der Spieler andauernd auf die entsprechende Taste drücken muss. Für Feuer (nicht Dauerfeuer!) oder zum springen ist der Befehl aber genau richtig.

keyboard_check_released verhält sich wie **keyboard_check_pressed**, führt aber den Befehl erst aus, wenn man die Taste wieder loslässt. Das eignet sich sehr gut für Abfragen der Escape-Taste und vergleichbaren Dingen. Hier ist es besser, den Befehl erst auszuführen, wenn der Spieler die Taste loslässt.

keyboard_check verhält sich ebenso, wie **keyboard_check_pressed**, aber mit dem Unterschied, das der Befehl nicht einmal, sondern dauernd ausgeführt wird, bis der Spieler die Taste loslässt. Das ist also der richtige Befehl, um das Objekt zu steuern und wäre auch der richtige Befehl für ein Dauerfeuer.

```
if (keyboard_check(vk_right))
{
    x += 4;
}

if (keyboard_check(vk_left))
{
    x -= 4;
}

if (keyboard_check(vk_down))
{
    y += 4;
}

if (keyboard_check(vk_up))
{
    y -= 4;
}
```

Der Code ist eigentlich sehr simpel. Wir fragen ab, ob gerade die entsprechende Taste (in den Klammern, hinter **keyboard_check**) gedrückt wird und ändern entsprechend über x und y unsere Richtung.

Gamepad Steuerung

So, wie man die Tastatur abfragen kann, kann man auch das Gamepad abfragen. Um dies tun zu können, gibt es den Befehl **gamepad_axis_value**. Damit fragt man ab, in welche Richtung ein Stick des Gamepads gedrückt wurde. Der Befehl ist wir folgt aufgebaut:

gamepad_axis_value(Gamepad Nummer, Index der Achse);

Da mehrere Gamepads an einem Computer hängen können, müssen wir uns eines davon herauspicken. Wenn ein Gamepad angeschlossen ist, hat es die Nummer 0. Diese Nummer werden wir auch verwenden. Bei den Achsen, bzw. beim Stick, haben wir die Möglichkeit, einen von zwei analogen Sticks oder das Digitalkreuz abzufragen. Im Beispiel verwende ich den linken Stick, der meistens für die Steuerung benutzt wird. Nach dem folgenden Schema kannst Du aber auch einen anderen Stick, das Digitalkreuz oder alle drei gleichzeitig verwenden.

Wenn wir nur die Abfrage für rechts verwenden würden, dann sähe der Code so aus:

```
if ((gamepad_axis_value(0, gp_axislh) > 0))
{
    x += 4;
}
```

Wenn Dein Gamepad angeschlossen ist und Du diesen Code verwendest, wird sich dein Objekt höchstwahrscheinlich nach rechts bewegen, ohne das Du den Stick benutzt. Das ist ein Problem, mit dem wir uns später befassen. Erst einmal widmen wir uns der Tatsache, dass wir hier doppelten Code generieren, nämlich für Tastatur und Gamepad. Da das sehr unschicklich ist, verkürzen wir die Sache. Der Code für alle vier Richtungen und beiden Steuergeräten sieht dann so aus:

```
if (keyboard_check(vk_right) || (gamepad_axis_value(0, gp_axislh) > 0))
{
    x += 4;
}

if (keyboard_check(vk_left) || (gamepad_axis_value(0, gp_axislh) < 0))
{
    x -= 4;
}

if (keyboard_check(vk_down) || (gamepad_axis_value(0, gp_axislv) > 0))
{
    y += 4;
}

if (keyboard_check(vk_up) || (gamepad_axis_value(0, gp_axislv) < 0))
{
    y -= 4;
}
```

Das sieht schon recht elegant aus. Wir fragen jeweils ab, ob die entsprechende Taste auf der Tastatur, oder (das ist das ||) die Richtung des Sticks gedrückt wurde. Die Abfrage ließe sich nun beliebig durch den zweiten Stick, dem Digitalkreuz oder den W, A, S und D-Tasten erweitern. Wichtig ist, dass wir mit **oder** und nicht mit **und** arbeiten, weil man sonst, Tastatur und Gamepad gleichzeitig benutzen muss.

Das sähe wirklich sehr albern aus.

Wenn Du den Code startest, merkst Du wahrscheinlich, dass das Objekt sich in irgendeine Richtung bewegt, obwohl Du nichts machst. Mein Objekt haut immer nach rechts oben ab und wünscht mir einen schönen Tag, sobald er den Raum verlässt. Der Grund ist sehr einfach: Die analogen Sticks sind viel zu empfindlich. Das heißt nicht, dass sie in Tränen ausbrechen, wenn man sie beleidigt oder anschreit, sondern das sie schon reagieren, obwohl man sie nicht drückt. Um das in den Griff zu bekommen, muss man eine sogenannte **Deadzone** vorgeben. Das ist ein Bereich, in dem das Signal des Gamepads noch als 0 wahrgenommen wird.

Um das tun zu können, gibt es den Befehl **gamepad_set_axis_deadzone**. Hiermit geben wir an, wie empfindlich ein bestimmtes Gamepad sein soll. Wir brauchen also erst das Pad, bei uns die Nummer 0, und den Wert. Der Wert liegt zwischen 0 und 1. Wenn man den Befehl nicht verwendet, ist die Zone bei 0 und was passiert, habe ich bereits beschrieben. Stellen wir den Wert auf 1, reagiert der Stick nicht mehr, weil 100% der Richtung als Deadzone angesehen wird. Die Gamemaker-Hilfe empfiehlt einen Wert von 0.05, also 5%. Meine Empfehlung lautet 0.2 (20%), weil manche Gamepads sehr empfindlich reagieren und eine sofortige Bewegung meist nicht erwünscht ist.

Den Code müssen wir jetzt nur noch im **Create-Event** des Objekts setzen:

```
gamepad_set_axis_deadzone(0, 0.2);
```

Das war es schon. Bei Speilstart verharrt das Objekt in völliger Ruhe, wie ein buddhistischer Mönch, sobald man aber die Achse oder die Tastatur drückt, geht es ab. So muss das sein!

In den meisten Spielen hat man mit dem Gamepad gewisse Vorteile, weil sich die Richtung präziser einstellen lässt. Abgesehen davon sollte man darauf achten, dass alle Steuergeräte möglichst gleichwertig sind und man nicht mit einem Gerät Vorteile erlangt. Auch hier kann man mit der Empfindlichkeit der Achsen ein wenig optimieren.

Date Created
8. Dezember 2016
Author
sven