Game Maker Studio: Die wichtigsten Befehle und Variablen

Events:
[create] | [step] | [alarm[0-9]] | [draw] | [drawGUI] | [collision] | [destroy]

Allgemeines:
randomize() | irandom() | get_timer() | room_speed | alarm[0-9] | screen_save_part()

Kollision:
collision_point() | collision_... | place_free() | place_meeting() | distance_to_object() | point_distance()

Bewegung:
motion_set() | move_contact_all() | xprevious/yprevious | xstart/ystart | path_start() | path_position | path_speed | point_direction()

Auf andere Objekte zugreifen:
id | with() | other | instance_number() | instance_create() | instance_destroy() | instance_find() | instance_nearest() | object_get_name() | object_index

Zeichnen:
draw_self() | draw_text() | draw_sprite() | draw_line() | draw_... | draw_set_... | make_color_rgb() | effect_create_above() | image_angle | image_index | image_speed

Input:
mouse_... | device_mouse_x(0) | device_mouse_check_button() | keyboard_check() | vk_left | vk_... | ord('A') | keyboard_check_pressed() | joystick_...

wichtig: Zum Auslesen von Buchstabentasten über den Befehl ord() muss der entsprechende Buchstabe als Großbuchstabe angegeben werden.
wichtig: Joystickfunktionen werden aktuell nur unter Windows unterstützt; eine Alternative ist die kostenlose Version von *Joy2Key*.

Views:
view_xview[] | view_wview[] | view_object[] | view_angle[] | view_enabled

wichtig: [drawGUI] zeichnet an die Bildschirmposition (d.h. unabhängig vom aktuellen view), [draw] dagegen an die Position im room!

Physik:
physics_apply_force() | physics_apply_impulse() | physics_apply_torque() | phy_,,

Game Maker Studio: Einige nützliche Formeln

Auf alle Instanzen im Raum einwirken

```
for (i = 0; i < instance_count; i += 1;) {
    with (instance_id[i]) {
        [...]
    }
}
```

Screen Wrap:

```
if (x<0 || x>room_width) x=room_width-x;
if (y<0 || y>room_height) y=room_height-y;
```

Test auf gerade/ungerade Zahlen

```
if (a&1) { [...] // wenn a eine ungerade Zahl ist
```

Objekt auf Kreisbahn bewegen

```
[step]
ang+=1;
if (ang>360) ang-=360;
x=tx+cos(degtorad(ang))*r; // tx=horiz. Zentrum des Kreises, r=Radius
y=ty+sin(degtorad(ang))*r; // ty=vert. Zentrum des Kreises, r=Radius
```

Objekt in Richtung eines Punktes drehen

```
dx=device_mouse_x(0)-(room_width/2);
dy=device_mouse_y(0)-(room_height/2);
ang = radtodeg(arctan2(dy,dx)); // gleich als Winkel anstatt Bogenmaß ausgeben
draw_sprite_ext(meinsprite,0,room_width/2,room_height/2, 1, 1, -ang, c_white, 1 );
```

Kollisionsabfrage nach allen Objekten eines Typs auf einer Linie

```
[step]
nearest=99999;
nearestid=noone;
with (obj_type) {
    if (collision_line(x1, y1, x2, y2, id, false, false)){
        dist=point_distance(x,y,other.x,other.y);
        if (dist< other.nearest) {
            other.nearest=dist;
            other.nearestid=id;
        [...]
    }
}
```