24 slots Inventory Tutorial

Door Joris van Leeuwen

In deze tutorial gaan we een inventory maken, met 24 sloten. De items in het inventory moet kunnen verplaatst, gewisseld, en verwijderd kunnen worden met de muis. Verscheidene items moeten samen één nieuw item kunnen vormen. Daarnaast gaan we ook een paar functies toevoegen die je vaak nodig hebt in een spel met een inventory.

Voor deze tutorial moet je al aardig wat ervaring hebben met GML. We gaan veel werken de FOR statement (die nog uitgelegd wordt in deze tutorial). Daarnaast gaan we ook veel werken met de functie variable_local_get(name) en de functie variable_local_set(name,value).

Als je klaar bent met deze tutorial kan je zelf al een aardige inventory in elkaar zetten.

Sprites aanmaken

Allereerst hebben we natuurlijk de sprites nodig. We gebruiken in deze tutorial maar 2 sprites.

We beginnen met de sprite voor de achtergrond van een slot (een vakje waar een item in kan zitten). Alle sprites uit deze tutorial moeten 32x32 worden.

→ Maak een sprite aan en noem deze *spr_slot*.
→ Maak een plaatje dat lijkt op de illustratie links van hier.
→ Zorg ervoor dat de sprite niet transparant wordt.

Nu hebben we nog plaatjes nodig voor alle items. We zouden nu voor iedere verschillende item een andere sprite kunnen gebruiken, maar aangezien de items geen animatie nodig hebben, kunnen we ze net zo goed allemaal in één sprite zetten. Dit zal ons later ook nog eens helpen bij het coderen.

→ Maak een sprite aan en noem deze *spr_item*
→ Zorg ervoor dat de:
- frame0 helemaal wit is (dit gaat dienen als een leeg itemslot).
- frame1 een groene gitaar is. 
- frame2 een blauwe gitaar is. 
- frame3 een gele gitaar is. 
- frame5 een rode gitaar is. 
- frame6 een bruine gitaar is. 
- frame7 een paarse gitaar is. 

→ Zorg ervoor dat de sprite transparant wordt.

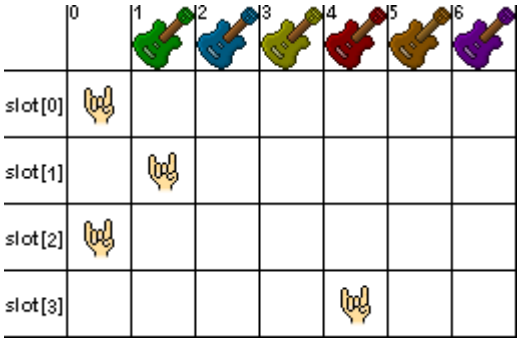
Zo, dat zijn alle items die we gaan gebruiken. Nu alle sprites klaar zijn, kunnen we beginnen met het maken van het variabel *slot[]*.

Het variabel *slot[]*

Wat we willen, is een inventory met 24 sloten. Elk van deze sloten moeten op zich een waarde hebben. Deze waarde wordt numeriek.

Zo kan bijvoorbeeld *slot[0]* een 0 zijn, wat staat voor *geen item*. En kan *slot[1]* bijvoorbeeld 3 zijn, wat staat voor *blauwe gitaar*. Zoals je ziet heeft de volgorde in de frames van *spr_item* te maken met de waarde van een *slot[]*.

Eigenlijk is *slot[0]* dus, zoals ik het zelf graag noem, een 2D variabel. Kijk maar een naar het volgende schema, van een inventory met 4 sloten.



Zoals je ziet heeft *slot[0]*, de waarde *0*. *0* staat voor niks, dus heeft het eerste slot op dit moment niks vast.

slot[1] heeft de waarde *1*, wat staat voor de groene gitaar. Het tweede slot heeft dus een groene gitaar vast, en dat is een feit dat we opslaan met het nummer *1*.

Op die manier onthouden we dus welke items in onze inventory zullen gaan komen, en die variabelen zullen komen in het enigste object dat we gaan gebruiken in deze tutorial. Het object *obj_controller*.

→ Maak een nieuw object aan, en noem deze *obj_controller*.

We gaan als eerste alle 24 *slot[]* variabelen (0 t/m 23) aantonen in het Creation event van de *obj_controller*. Dit lijkt voor sommige programmeurs misschien als veel werk, maar wij gaan het dit keer met een handige methode aanpakken: de FOR statement.

Voor de programmeurs die de FOR statement nog niet kennen:

De For statement wordt gebruikt voor een code die een aantal keer herhaald moet worden, en in deze herhaling rekening houdt met de hoeveelste keer dat de code al herhaald is. De algemene opbouw van een FOR statement is als volgt:

For (*var=0; var<waarde; var+=1){*

Code;

}

Hier kan de *var* bijvoorbeeld *i* zijn. Je zegt dus als eerste dat (bijvoorbeeld) *i=0;*. Vervolgens geef je aan onder welke voorwaarde de *Code* nog een keer herhaald moet worden. Dat kan dus bijvoorbeeld *i<4;* zijn. Tot slot geef je aan wat er na iedere herhaling moet gebeuren, en dat is waar je dan bijvoorbeeld *i+=1* zou kunnen neerzetten. Op die manier herhaald hij de *Code* 4 keer, en kan je in de *Code* ook nog eens met de variabel *i* werken (die overigens na het script wordt vergeten).

→ Maak een nieuw event aan: Creation event.

→ Voeg een code toe aan het Creation event:

```
{
    for (i=0; i<24; i+=1){
        variable_local_set("slot["+string(i)+"]",round(random(6)));
    }
}
```

Zoals je ziet maken we gebruik van de FOR statement. De *Code* wordt 24 keer herhaald.

In de FOR zelf, maken we gebruik van de functie *variable_local_set(name, value)*. Deze functie laat ons de naam van een variabel aanpassen met *strings handling functions*.
Door bij de naam van de variabel *"slot["+string(i)+"]"* in te vullen, maken we door de FOR statement de 24 *slot[]* variabelen aan. Binnen de *[* en de *]* komt nu het een getal op basis van de *i*. We hoefden ze dus niet zelf omstebeurt aan te maken.
Door bij de waarde *round(random(6))* in te vullen, krijgt ieder slot een willekeurige item *random(6)* zorgt voor een willekeurig getal tussen 0 en 6, *round()* zorgt ervoor dat dit een heel getal wordt).

Het tekenen van de sloten

Nu we de sloten hebben aangemaakt, willen we eigenlijk ook wel de uitkomst zien. We gaan hiervoor veel werken met het Draw event.

→ Maak een nieuw event aan: Draw event.

Wat we willen is dat de sloten van de inventory in 6 rijen komen te staan. Deze rijen hebben dan dus 4 sloten van links naar rechts.

Deze sloten moeten dan gaan optellen van links naar rechts, van boven naar beneden.

Links van hier zie je een illustratie met daarin een lege inventory, met op ieder vakje het nummer van het slot.

Om deze opbouw te bereiken, moeten we 6 keer een code herhalen die 4 keer van links naar rechts een vakje tekent. Oftewel, we gaan een FOR statement gebruiken IN een andere FOR statement.

→ Voeg een code toe aan het Draw event:

```
{
    //draw slots
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            draw_sprite(spr_slot,0,x+32*b,y+32*i);
        }
    }
}
```

Zoals je ziet, hebben we hier gebruik gemaakt van een FOR statement, in een andere FOR statement.

In de twee FOR's, hebben we gebruik gemaakt van de functie *draw_sprite(sprite, image, x, y)*. We tekenen de sprite *spr_slot*, met uiteraard, frame0. Het moeilijkste deel van deze code zijn de coördinaten waar de sprite getekend moet worden.

De x waarde hangt af van in welke horizontale herhalingstap de FOR zit. Bij de eerste stap, moet de sprite op de zelfde x waarde komen als de x waarde van de obj_controller zelf. Bij de tweede stap, moet de sprite 32 pixels (de breedte van het plaatje) verder getekend worden. De x waarde van het plaatje is dus *x+32*aantal horizontale stappen*, en het aantal stappen is in dit geval opgeslagen in *b*. Dus uiteindelijk is de x waarde van het plaatje *x+32*b*.

De y waarde hangt af van in welke verticale herhalingstap de FOR zit. De theorie hierachter is vergelijkbaar met de theorie over de x waarde. Het enigste verschil is dat je nu met de variabel *i* moet werken in plaats van met de *b*. de y waarde van een plaatje is dus *y+32*i*.

Voordat je het "spel" test, moet je natuurlijk wel even een room aanmaken.

→ Maak een nieuwe room aan, en noem deze *room_inventory*.

→ Plaats een instantie van *obj_controller* een beetje links bovenin de room.

Als je het spel nu test, zie je dat er 24 vakjes getekend worden, in de volgorde die we wilden.

Het tekenen van de items

Nu we de sloten hebben getekend, kunnen we daar overheen de items tekenen, op basis van de *slot[]* variabelen hun waardes.

→ Open de code in het Draw event en vervang:

```
{
    //draw slots and items
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            draw_sprite(spr_slot,0,x+32*b,y+32*i);

            draw_sprite(spr_item,variable_local_get("slot["+string(i*4+b)+
            "]"),x+32*b,y+32*i);
        }
    }
}
```

Weer maken we hier gebruik van de functie *draw_sprite(sprite, image, x, y)*. We tekenen hier de sprite *spr_item*.

De frame die hier getekend moet worden hangt af van de waarde van het *slot[]*, en welk *slot[]* hangt af van zowel de horizontale als van de verticale FOR. We maken gebruik van de functie *variable_local_get(name)*, omdat je hiermee met *string handling function* kunt werken. We vullen in *"slot["+string(i*4+b)+"]"*. We tellen de *b* (horizontale FOR), op bij de *i* (verticale FOR) die door 4 is vermenigvuldigd. Deze methode wordt misschien duidelijker als je terug kijkt naar de illustratie van de inventory met nummers op de vorige pagina. In de eerste kolom, is de waarde *i* gelijk aan 0 en is de telt de *b* normaal op. In de tweede kolom, is de waarde *i* gelijk aan 1 en wordt deze *1* in onze methode vermenigvuldigd met *4*, waardoor de eerste kolom meegeteld wordt. De *b* telt ook hier weer normaal op, en hierdoor, als je de *i*4+b* gebruikt, krijg je het juiste nummer van het slot. Hierdoor zoekt de functie *variable_local_get(name)* naar de juiste *slot[]* variabel, en wordt de waarde teruggehaald die het juiste item laat tekenen.

De theorie achter de coördinaten van de items die we tekenen, is hetzelfde als met het tekenen van de sloten. Oftewel *x+32*b,y+32*i*.

Als je het spel nu test, zie je dat er allemaal items worden getekend in de sloten. Voor het makkelijkere testen, gaan we even een optie maken om het spel te herstarten.

→ Maak een nieuw event aan: Key press R event.

→ Voeg een code toe aan de Key press R event:

```
{
    game_restart();
}
```

Hierdoor kunnen we snel en makkelijk met een druk op de R knop, het spel laten herstarten.

Items selecteren

Nu we de items hebben aangemaakt, willen we er eigenlijk ook wat mee doen. Uiteindelijk willen we de items kunnen verslepen van slot naar slot. Voordat we kunnen beginnen aan het verslepen van items, moeten we eerst een soort selecteer systeem maken.

Om op te slaan welk slot geselecteerd is, maken we een nieuwe variabel aan: *selected*. Deze variabel wordt numeriek en het nummer dat is opgeslagen zal weer gaan geven welk slotnummer is geselecteerd. *-1* staat voor *geen selectie*.

→ Open de code in het Creation event en vervang:

```
{
    for (i=0; i<24; i+=1){
        variable_local_set("slot["+string(i)+"]",round(random(6)));
    }

    selected = -1;
}
```

We zetten de *selected* naar *-1*, aangezien we niet willen dat er een slot is geselecteerd wanneer we het spel beginnen.

Wanneer we met de linker muisknop op een slot klikken, zal het slot geselecteerd moeten worden. We moeten dus een nieuw event aanmaken: *Global Mouse left pressed*. In dit event zal getest moeten worden in welk slot er geklikt is, en op basis daarvan de variabel *selected* verzetten.

→ Maak een nieuw event aan: Global Mouse left pressed.

→ Voeg een code toe aan het Global Mouse left pressed event:

```
{
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            if (mouse_x > x+32*b && mouse_x < x+32+32*b && mouse_y >
            y+32*i && mouse_y < y+32+32*i){
                selected = i*4+b;
            }
        }
    }
}
```

Wat we bereiken met deze code, is dat de computer in 24 vlakken checked of de muis daarbinnen heeft geklikt. Zo ja, dan wordt het nummer van het vlak waarin is geklikt opgeslagen in *selected*.

Uiteraard willen we een leuk effect hebben, wat ons laat zien welk *slot[]* we hebben geselecteerd. Dit gaan we helemaal in-code doen. We willen dat een geselecteerd slot wit gaat gloeien. Dit gaan we bereiken door er een wit transparant vierkantje overheen te tekenen. De transparantheid van dit witte vierkantje gaat op en neer met behulp van *sin()*. Omdat we een *sin()* gaan gebruiken, hebben we er een variabel voor nodig. Deze noemen we *draw_selected*. Hiermee kunnen we met behulp van de *sin()* berekenen hoe transparant het witte vierkantje moet zijn.

→ Open de code in het Creation event en vervang:

```
{
    for (i=0; i<24; i+=1){
        variable_local_set("slot["+string(i)+"]",round(random(6)));
    }

    selected = -1;
    draw_selected = 0;
}
```

→ Open de code in het Draw event en vervang:

```
{
    //draw slots and items
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            draw_sprite(spr_slot,0,x+32*b,y+32*i);

            if (selected == i*4+b){
                draw_selected += 0.3;
                draw_sprite_ext(spr_item,variable_local_get("slot["+stri
                ng(i*4+b)+"]"),x+32*b,y+32*i,1,1,0,c_white,0.8+0.2*sin(d
                raw_selected));
                draw_set_color(c_white);
                draw_set_alpha(0.1+0.3*sin(draw_selected));

                draw_rectangle(x+32*b,y+32*i,x+32*b+32,y+32*i+32,false);
                draw_set_alpha(1);
            }else{
                draw_sprite(spr_item,variable_local_get("slot["+string(i
                *4+b)+"]"),x+32*b,y+32*i);
            }
        }
    }
}
```

Ten eerste wordt het slot zelf getekend (die hadden we al eerder neergezet). Vervolgens wordt er met de regel *if (selected == i*4+b)* afgevraagd of de FOR stap waarin de code op dat moment zit toevallig geselecteerd is. Als hij geselecteerd is, dan wordt de *draw_selected* verhoogt, en wordt het item getekend. Vervolgens wordt de tekenkleur naar wit gezet, en de *alpha* naar *0.1+0.3*sin(draw_selected)*. Hierdoor verhoogt en verlaagt de *alpha* met een golfachtige beweging (stuk mooier dan gewoon op en neer). Vervolgens wordt er een vierkant getekend, met coördinaten op basis van de FOR stap waarin de code zich begeeft. Tot slot wordt de *alpha* weer naar *1* gezet (voor het geval dat je ergens anders nog iets gaat tekenen). Als het slot niet geselecteerd zou zijn, wordt er simpelweg het item getekend.

Zo, nu wordt er duidelijk laten zien welk slot we geselecteerd hebben, met nog een leuk effect ook!

Items verslepen

Nu we sloten kunnen selecteren, is het tijd voor het verslepen van items. Wanneer je klikt op een item, moet een transparant plaatje van het item je muis volgend. Wanneer je de muis loslaat op een ander slot, verdwijnt het transparante item, en wordt de waarde van het geselecteerde slot geruild met de waarde van het slot waarop je de muis heb losgelaten.

Hiervoor hebben we een nieuwe variabel nodig, die opslaat of een item wordt gesleept: *drag*.

→ Open de code in het Creation event en vervang:

```
{
    for (i=0; i<24; i+=1){
        variable_local_set("slot["+string(i)+"]",round(random(6)));
    }

    selected = -1;
    draw_selected = 0;
    drag = false;
}
```

Nu moeten we het Global Mouse left pressed event zo aanpassen, dat *drag* de waarde *true* krijgt als in het geklikte slot een item zit. We willen natuurlijk niet dat je een leeg slot kunt verslepen (hoe kan je een item verslepen als die er niet is). Daarnaast gaan we ook de *draw_selected* naar *0* zetten, zodat het gloeiende witte vierkantje weer overnieuw begint met gloeien.

→ Open de code in het Global Mouse left pressed event en vervang:

```
{
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            if (mouse_x > x+32*b && mouse_x < x+32+32*b && mouse_y >
            y+32*i && mouse_y < y+32+32*i){
                selected = i*4+b;
                draw_selected = 0;
                if (variable_local_get("slot["+string(i*4+b)+"]") != 0){
                    drag = true;
                }
            }
        }
    }
}
```

Zoals je ziet hebben we de regel *draw_selected = 0* toegevoegd om de gloeiende animatie overnieuw te laten beginnen.

Daarnaast wordt er gevraagd of het slot waarop we geklikt hebben een item bevat (oftewel, of het slot zijn waarde geen *0* is), met de regel *if (variable_local_get("slot["+string(i*4+b)+"]") != 0)*. Als het slot een item bevat, wordt *drag* naar *true* gezet (we kunnen beginnen met verslepen).

Nu we weten wanneer een item gesleept wordt, kunnen we een plaatje van het slepende item, transparant de muis laten volgen op basis daarvan. Hierdoor krijgt de gebruiken een stuk beter het idee dat hij het item aan het verslepen is.

→ Open de code in het Draw event en vervang:

```
{
    //draw slots and items
    for (i=0; i<6; i+=1){
        for (b=0; b<4; b+=1){
            draw_sprite(spr_slot,0,x+32*b,y+32*i);

            if (selected == i*4+b){
                draw_selected += 0.3;
                draw_sprite_ext(spr_item,variable_local_get("slot["+stri
                ng(i*4+b)+"]"),x+32*b,y+32*i,1,1,0,c_white,0.8+0.2*sin(d
                raw_selected));
                draw_set_color(c_white);
                draw_set_alpha(0.1+0.3*sin(draw_selected));

                draw_rectangle(x+32*b,y+32*i,x+32*b+32,y+32*i+32,false);
                draw_set_alpha(1);
            }else{
                draw_sprite(spr_item,variable_local_get("slot["+string(i
                *4+b)+"]"),x+32*b,y+32*i);
            }
        }
    }

    //draw dragged item
    if (drag){
        draw_sprite_ext(spr_item,variable_local_get("slot["+string(selecte
        d)+"]"),mouse_x-16,mouse_y-16,1,1,0,c_white,0.7);
    }
}
```

Het nieuwe deel van de code staat onder de *//draw dragged item*.

Er wordt eerst gevraagd of er iets versleept wordt met de regel *if (drag)*. Als dat het geval is, dat wordt er een plaatje getekend met de functie *draw_sprite_ext*, waardoor we het plaatje lichtelijk transparant kunnen maken. De frame die van *spr_item* getekend moet worden, is gelijk aan de waarde van het slot dat geselecteerd is. Dus hiervoor kunnen we de regel *variable_local_get("slot["+string(selected)+"]")* voor gebruiken om de waarde van het geselecteerde *slot[]* op te vragen. De positie van het plaatje is de coördinaten van de muis-16, aangezien we willen dat het plaatje gecentreerd wordt op de muis, en de plaatjes allemaal 32x32 zijn.

Nu kunnen we zien welk item versleept wordt, en dat die versleept wordt. We kunnen echter nog niks echt verslepen. Daar gaan we nu verandering in brengen. We gaan gebruik maken van een nieuw event: *Global Mouse left released*. Hier gaan we testen in welk slot de muis zijn linkerknop loslaat, en op basis daarvan gaan we de twee waardes van het geselecteerde slot, en van het slot waar de muis op is losgelaten verwisselen. Stel dat je *slot[0]* geselecteerd hebt, die de waarde *3* bevat, en vervolgens de muis loslaat op *slot[2]*, die de waarde *0* bevat. Dan zou *slot[0]* de waarde *0* krijgen, en *slot[2]* de waarde *3*.

→ Maak een nieuw event aan: Global Mouse left released.
→ Voeg een code toe aan het Global Mouse left released event:

```
{
    if (drag){
        for (i=0; i<6; i+=1){
            for (b=0; b<4; b+=1){
                if (mouse_x > x+32*b && mouse_x < x+32+32*b && mouse_y >
                 y+32*i && mouse_y < y+32+32*i){
                    var a;
                    a = variable_local_get("slot["+string(i*4+b)+"]");
                    variable_local_set("slot["+string(i*4+b)+"]",
                     variable_local_get("slot["+string(selected)+"]"));
                    variable_local_set("slot["+string(selected)+"]", a);
                }
            }
        }
    }

    drag = false;
    selected = -1;
}
```

Deze code lijkt voor sommige mensen misschien een beetje veel tegelijk, en dat is het misschien ook wel. Daarom gaan we het stap voor stap doornemen.

Ten eerste wordt er gevraagd of er iets werd versleept. Als dit het geval was, wordt er met de twee FOR's gevraagd in welk *slot[]* de muis werd losgelaten. Dat trucje is te vergelijken met hetgeen wat we gemaakt hadden in het Global Mouse left pressed event om te checken welk slot geselecteerd werd.

Als de FOR in de stap zit van het slot waarin de muis werd losgelaten, komt onze wisseltruc in werking. Ten eerste maken we de variabel *a* aan. Deze variabel slaat de waarde van het slot waarin de muis is losgelaten op met de regel *a = variable_local_get("slot["+string(i*4+b)+"]")*. Vervolgens wordt de waarde van het slot waarop we de muis hadden losgelaten omgezet naar de waarde van het geselecteerde slot met de regel *variable_local_set("slot["+string(i*4+b)+"]", variable_local_get("slot["+string(selected)+"]"));*. Omdat we de nu verloren waarde van het slot waarop we de muis hadden losgelaten hadden opgeslagen in de variabel *a*, kunnen we nu de waarde van het geselecteerde slot omzetten naar de variabel *a*, met de regel *variable_local_set("slot["+string(selected)+"]", a)*. Hiermee zijn de waardes in de sloten verwisseld.

Tot slot zetten we de variabel *drag* naar de waarde *false*, en de variabel *selected* naar *-1*, aangezien we niet meer verslepen en niets meer geselecteerd willen hebben.

Items weggooien

Nu we items kunnen verslepen, kunnen we ook beginnen aan het weggooien van items. Dit doe je door een item te selecteren en vervolgens buiten de inventory te slepen. Dit is in codes niet zo super moeilijk te bereiken, wat we doen is bij het Global Mouse left released te vragen of de coördinaten van de muis buiten de inventory zitten. Als dat het geval is wordt het slot dat geselecteerd was naar waarde *0* gezet.

→ Open de code in het Global Mouse left released event en vervang:

```
{
    if (drag){
        for (i=0; i<6; i+=1){
            for (b=0; b<4; b+=1){
                if (mouse_x > x+32*b && mouse_x < x+32+32*b && mouse_y >
                 y+32*i && mouse_y < y+32+32*i){
                    var a;
                    a = variable_local_get("slot["+string(i*4+b)+"]");
                    variable_local_set("slot["+string(i*4+b)+"]",
                     variable_local_get("slot["+string(selected)+"]"));
                    variable_local_set("slot["+string(selected)+"]", a);
                }
            }
        }
    }

    if (mouse_x < x || mouse_x > x+128 || mouse_y < y || mouse_y > y+182){
        variable_local_set("slot["+string(selected)+"]", 0);
    }

    drag = false;
    selected = -1;
}
```

Met de regel *if (mouse_x < x || mouse_x > x+128 || mouse_y < y || mouse_y > y+182)* testen we of de coördinaten van de muis misschien buiten de inventory zit. Als dat het geval is, geven we met de regel *variable_local_set("slot["+string(selected)+"]", 0)*, het *slot[]* dat geselecteerd was de waarde *0*. Hierdoor is het voormalige item in het geselecteerde slot weggegooid.

Items combineren

Nu hebben we al een aardige inventory in elkaar gezet. We kunnen items verslepen, items verwisselen, en ook al items weggooien.

Wat je bij veel spellen hebt, is dat een item bij een ander item een heel nieuw item wordt. Denk maar aan bijvoorbeeld blaadjes en kersen. Als je deze twee bij elkaar doet krijg je dan salade in het slot waarop je de muis hebt losgelaten, en het geselecteerde slot wordt leeg.

In onze inventory, gaan we de kleuren van de gitaren laten mengen met elkaar. Het mengen gaat met de volgende gitaren:
Blauw + Geel → Groen Geel + Rood → Paars Rood + Geel → Oranje/Bruin

Voor het mengen van de gitaren gaan we een script maken. Dit script heeft drie argumenten nodig: het ene nummer van de meng items, het andere nummer van de meng items, en het nummer van het item dat eruit komt als de twee items gemengd zijn. Dit script zetten we neer in het Mouse Release left button event, en het gaat kijken of de waarde van het geselecteerde slot overeenkomt met het eerste of tweede mengitem, en of het slot waarop de muis was losgelaten overeenkomt met het eerste of het tweede mengitem. Als dat zo is, dan past het script de waardes van de sloten aan, en geeft het een *false* terug. Als het geselecteerde slot, en het slot waarop de muis was losgelaten, niet de waardes van de mengitems hebben, dan geeft het script een *true* terug (de reden van het terug geven van de *true* en *false* wordt later duidelijk gemaakt). Op deze manier kunnen we dit script in een IF statement zetten, die uitmaakt of de items moeten verwisseld worden of niet (als we de items mengen hoeven we ze natuurlijk niet meer te mengen!).

→ Maak een nieuw script aan en noem deze *scr_item_combiner*.

→ Open de *scr_item_combiner* en vervang:

```
{
    var selected_slot, released_slot;
    selected_slot = variable_local_get("slot["+string(selected)+"]");
    released_slot = variable_local_get("slot["+string(i*4+b)+"]");

    if (selected_slot = argument0 && released_slot = argument1){
        variable_local_set("slot["+string(selected)+"]", 0);
        variable_local_set("slot["+string(i*4+b)+"]", argument2);
        return (false);
        exit;
    }else if (selected_slot = argument1 && released_slot = argument0){
        variable_local_set("slot["+string(selected)+"]", 0);
        variable_local_set("slot["+string(i*4+b)+"]", argument2);
        return (false);
        exit;
    }else{
        return (true);
    }
}
```

Dus, *argument0* staat voor het ene mengitem, *argument1* staat voor het andere meng item, en *argument2* staat voor het item dat eruit komt.

Zoals je ziet beginnen we het script met het benoemen van de variabelen *selected_slot* en *released_slot*, naar de waardes van het desbetreffende slot. Dit doen we zodat we niet overdreven lange regels hoeven te schrijven in de rest van het script.

Vervolgens wordt er gevraagd of het geselecteerde slot overeen komt met het ene meng item en het slot waarop de muis was losgelaten overeen komt met het andere meng item met de regel *if (selected_slot = argument0 && released_slot = argument1)*. Als dat het geval is, wordt de waarde van het geselecteerde slot naar *0* gezet met de regel *variable_local_set("slot["+string(selected)+"]", 0)*, en de waarde van het slot waarop de muis was losgelaten naar de waarde van het item dat uit de mengitems komt met de regel *variable_local_set("slot["+string(i*4+b)+"]", argument2)*. Vervolgens geeft het script *false* terug, en negeert het de rest van het script aangezien het in dat geval al klaar zou zijn.

Als dit niet het geval zou zijn, wordt er gevraagd of de mengitems misschien andersom liggen, met de regel *else if (selected_slot = argument1 && released_slot = argument0)*. Als dit het geval zou zijn worden dezelfde acties uitgevoerd als in het eerste geval.

Als beide mogelijkheden niet kloppen, geeft het script *true* terug.

Nu we dat script gemaakt hebben, kunnen we ermee gaan werken in het Global Mouse left released event.

→ Open de code in het Global Mouse left released event en vervang:

```
{
    if (drag){
        for (i=0; i<6; i+=1){
            for (b=0; b<4; b+=1){
                if (mouse_x > x+32*b && mouse_x < x+32+32*b && mouse_y >
                y+32*i && mouse_y < y+32+32*i){

                    if (script_execute(scr_item_combiner,2,3,1) &&
                    script_execute(scr_item_combiner,4,3,5) &&
                    script_execute(scr_item_combiner,4,2,6)){

                        var a;
                        a = variable_local_get("slot["+string(i*4+b)+"]");
                        variable_local_set("slot["+string(i*4+b)+"]",
                        variable_local_get("slot["+string(selected)+"]"));
                        variable_local_set("slot["+string(selected)+"]",
                        a);
                    }
                }
            }
        }
    }

    if (mouse_x < x || mouse_x > x+128 || mouse_y < y || mouse_y > y+182){
        variable_local_set("slot["+string(selected)+"]", 0);
    }


    drag = false;
    selected = -1;
}
```

Zoals je ziet, wordt er, voor dat de sloten worden verwisseld, gevraagd of er iets gemaakt moet worden met de regel *if (script_execute(scr_item_combiner,2,3,1) && script_execute(scr_item_combiner,4,3,5) && script_execute(scr_item_combiner,4,2,6))*. Hier staat het script met de argumenten *(2,3,1)* voor blauw+geel wordt groen, *(4,3,6)* voor rood+ geel wordt bruin, en *(4,2,6)* voor rood+blauw wordt paars. Als de items gemengd kunnen worden, dan komt er uit een van de scripts een *false*, en wordt de wisseltruc voor het wissel van de items niet uitgevoerd. Als er echter geen enkele menging mogelijk is, komt de wisseltruc in werking en worden de waardes van het geselecteerde slot en van het slot waarop de muis is losgelaten gewisseld. Om deze rede lieten we het script een *false* of een *true* teruggeven.

Voor iedere nieuwe combinatie van items hoeven we alleen maar een *script_execute(scr_item_combiner,item1,item2,item_new)* neer te zetten in de IF statement, en we zijn klaar!

Dat was de tutorial! Onze inventory is klaar. We kunnen items verplaatsen, wisselen, weggooien, en zelf combineren!

Als mensen nog vragen hebben en er echt niet zelf uit kunnen komen, suggesties hebben voor betere codes, of misschien gewoon hun mening willen uiten, kunnen zij mij via www.game-maker.nl bereiken bij de naam Urban Joris.

Extra functie: tellen van items

Misschien heb je het wel eens nodig om het aantal items in de inventory te checken. De hoeveelheid item in een inventory als deze is terug te halen met het volgende script:

```
var num_item;
num_item = 0;
for (i=0; i<23; i+=1){
    if (variable_local_get("slot["+string(i)+"]") != 0){
        num_item += 1;
    }
}
return (num_item);
```

Eerst wordt de variabel *num_item* aangetoond, en verzet naar de waarde *0*. Vervolgens wordt er met behulp van een FOR statement alle sloten langs gegaan of ze misschien een waarde hebben anders dan *0* met behulp van de regel *if (variable_local_get("slot["+string(i)+"]") != 0)*. Iedere keer dat een itemslot een item bevat, telt die een *1* op bij de *num_item* met de regel *num_item += 1*.

Als je dus in het vervolg ergens het aantal items in de inventory nodig heb, kun je het ophalen met dit script.