Scripts schreiben

Konstruktion

Grundkonstruktion

Jedes Script hat ein 'INIT'-Block und ein 'DO'-Block. Der 'INIT'-Block wird nur einmal bearbeitet, nämlich am Anfang. Danach wird automatisch der 'Do'-Block ausgeführt. Solange das Spiel läuft und kein anderes Script aufgerufen wird, wird die ganze zeit der 'Do'-Block ausgeführt. Zwischen den '#' werden die Anweisungen geschrieben.

Die Böcke werden wie folgt erstellt:

```
'
#INIT

#

#Do

#
'
```

Syntax

Eine Anweisung wird immer in eine Zeile geschrieben und mit einem Semikolon ';' abgeschlossen.

Variablen

Variable-typen

Es gibt 'bool', 'int' und 'string';

Bool kann entweder true oder false annehmen.
Int kann alle geraden Zahlen annehmen.
String kann Zeichenketten speichern.

Eine Variable erstellen

Man schreibt den typ, den Namen der Variable, ein Istgleichzeichen und den Wert der Variable. Bsp.: 'string text = hallo;'.

Eine Variable einen Wert zuweisen

Man schreibt den Namen der Variable, hängt '.value' an, schreibt ein Istgleichzeichen und den Wert. Der Wert kann ein beliebiger sein (1) oder von deiner anderen Variable übernommen (2). Hierbei schreibt man den Namen der Variable und hängt '.value' an den Namen an.

Bsp(1).: 'text.value = ein anderer Text;'
Bsp(2).: 'text.value = text2.value;'

Ein Objekt ansprechen

Objekt-Eigenschaften verändern

Durch den Befehl 'object(string objectname)' kann man jedes Objekt, das sich auf der Map befindet, ansprechen. Danach folgt die Eigenschaft die man ändern möche.

Bsp.: object(Baum).mapX = 20;

Natürlich kann man auch aus einer bereits erstellten Variable einen Wert geben.

Bsp.: object(Baum).mapX = BaumX.value;

Liste der Eigenschaften eines Objekts

- mapX
- mapY
- width
- height
- present
- isColl
- isAnimated

Objekt-Methoden verwenden

Ein Objekt hat auch Methoden, die es erlauben, das Objekt zu animieren oder zu bewegen. Die Methoden werden wie die Eigenschaften an das Objekt angehängt.
In der eckigen Klammer werden die Parameter für die Methode geschrieben.

Bsp.: object(Auto).move[];

Objekt-Methoden

- **mapCol [ ]**

Prüft, ob das Objekt mit der Objektbegrenzung gegen den Maprand stößt. Wenn das Objekt dies tut, wird es automatisch gestopt.

- **create [string filename, int mapX, int mapY]**

Erstellt ein neues Objekt mit den folgenden Parametern.

- **loadAnimation [string filename, bool loop, int intervall]**

Läd eine Animations-Datei in das Objekt. Der Parameter 'loop' lässt bei True immer die Animation durchlaufen, bei Fales stopt sie nach einem Durchlauf. Der Intervall besagt, wie schnell die Bilder in Millisekunden wechseln sollen.

Objekt-Bedingungnen

Um Objekt-Bedungunen zu prüfen, muss man sie in die If-Anweisung schreiben.
Die meisten Bedingungen geben einen Bool zurück.
Man vergleicht also wie folgt eine Bedingung:

Bsp.:
```
'
if (object( auto ).touch [ baum ] == true)
{

}
'
```

- **object(objectname).touch [objectname2/eventname]**

Prüft nach, ob das Objekt die Begrenzung eines anderen Objekts oder eines Events überschreitet.

Ein Event ansprechen

Man sowohl Objekte als auch Events ansprechen. Dies tut man, indem man 'event' und in der danachfolgenden Klammer den Namen des Events schreibt.

Bsp.: 'event(eventname)'

Event-Eigenschaften verändern

Um eine Eigenschaft zu verändern, schreibt man einfach einen Punkt und dann die darauffolgende Eigenschaft.

Bsp.: 'event(ev).activ = false;'

Liste der Eigenschaften

- mapX
- mapY
- width
- height
- activ

Event-Methoden

Events können wärend dem Spiel ein und ausgschaltet werden. Um dies und noch andere Ereignisse in das Spiel einzbauen, gibt es Methoden dazu. Die Methode wird, wie die Eigenschaft, mit einem Punkt angehängt.

Bsp .: 'event(ev).actived[ ];'

- **actived [ ]**

Aktiviert das Event.

- **deactived [ ]**

Dektiviert das Event.

Event-Bedingungen

Um zu Prüfen, ob ein Event ein Objekt berührt hat oder eine andere Bedingung erfüllt ist, benutzt man Event-Bedingungen.

- **event(eventname).touch [objectname/eventname]**

Prüft nach, ob das Event die Begrenzung eines anderen Objekts oder eines Events überschreitet.

If-Anweisung

Eine If-Anweisung ist dazu da, Ereignisse zu prüfen wie Beispielsweise die Berührung eines Objektes mit einem Event. Wenn dies der Wahrheit entspricht, werden die folgenden Anweisungen in den geschweiften Klammern ausgeführt.

Eine If-Anweisung muss wie folgt geschrieben werden:

'
if (*Bedingung*)
{

}
'

If-else-Anweisung

Wenn nach dem If-Block noch ein else-Block folgt, wird dieser dann ausgeführt, wenn die Überprüfung nicht der Wahrheit entspricht.

Eine If-else-Anweisung muss wie folgt geschrieben werden:

```
if ( Bedingung )
{

}
else
{

}
```

Variablen vergleichen

Es gibt 3 Möglichkeiten Variablen zu prüfen.

1.) Mit '==' prüft man, ob der Wert links davon gleich den Wert rechts davon ist.
2.) Mit '>' prüft man, ob der Wert links davon größer ist, als der Wert rechts davon.
3.) Mit '<' prüft man, ob der Wert links davon kleiner ist, als der Wert rechts davon.

Die beiden letzten Möglichkeiten können nur mit dem variablen-typ 'int' genommen werden. Ein Vergleich könnte so aussehen:

Bsp.:

```
if (zahl.value < 8)
{

}
```