1 Ein richtiges Spielprojekt

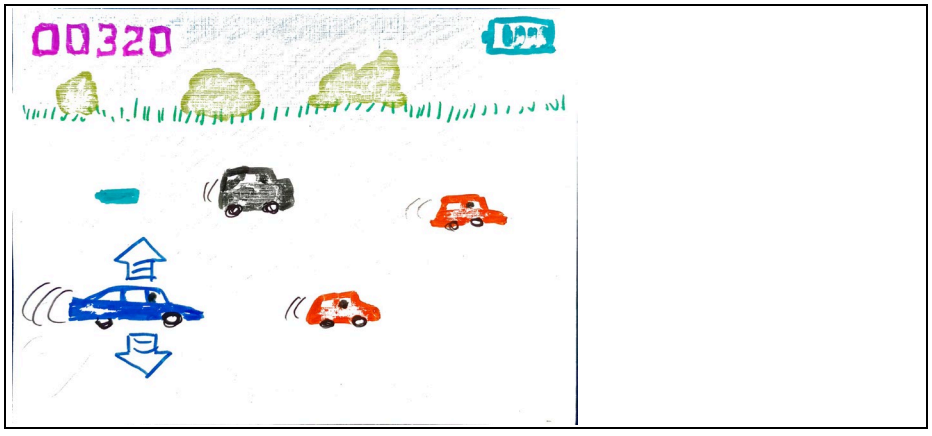
Die bisherigen Scratch-Spielereien in diesem Buch lassen sich kaum als richtige Spiele bezeichnen. Games müssen spannend, spaßig und cool sein. Herausforderung angenommen: Hier kommt Agent X auf geheimer Mission in seinem flotten, blauen Sportwagen!

1.1 Was macht ein gutes Spiel aus?

Einfache Spiele sind nett, werden aber schnell langweilig. Erfolgreich sind auf Dauer nur Spiele, die Spaß machen, spannend sind und voller cooler Effekte. Spiele brauchen einen Punktezähler, Levels, Power-ups usw. Je umfangreicher ein Spiel, desto wichtiger ist es, dass es gut durchdacht aufgebaut ist. Sonst verzettelst du dich. Es wird schwerer, Fehler zu finden oder das Programm zu erweitern.

In diesem Kapitel zeige ich dir ein relativ kompliziertes Spiel, das gerade noch gut mit Scratch umsetzbar ist. Alles, was komplizierter ist, wird dann mit Godot erledigt.

Jedes Spielprojekt fängt an mit einer Idee. Am besten zeichnest du sie grob auf ein Blatt Papier. Etwa so:



Die Skizze zeigt Agent X, der einen äußerst dringenden Termin in seinem Geheimdienst-Hauptquartier hat, und die anderen, langsameren Verkehrsteilnehmer, die ärgerliche Hindernisse sind. Als Geheimagent muss sich Herr X natürlich nicht ans Tempolimit halten. Er darf und muss alle

anderen Autos überholen. Hinzu kommt, dass Agent X ein Elektroauto fährt. In der Bildschirmecke oben rechts ist der Ladezustand der Batterie zu sehen. Ist sie leer, bleibt das Auto stehen, und das Spiel ist verloren. Zum Glück liegen unterwegs ab und an Batterien auf der Straße, die von einer coolen Vorrichtung an seinem blauen Sportwagen eingesammelt werden und die Autobatterie wieder aufladen.

Letztlich bewegen sich die anderen Autos von links nach rechts und die Spieler steuern den Sportwagen in vertikaler Richtung mit den Pfeiltasten, um an ihnen vorbeizukommen und Batterien einzusammeln. Um die Illusion einer schnellen Fahrt zu erzeugen, gibt es am oberen Bildschirmrand Büsche und auf der Straße eine gestrichelte Mittellinie, die sich in entgegengesetzter Richtung der Autos von rechts nach links bewegen, schneller als die Autos, da diese ja fahren.

Zu guter Letzt gibt es links oben einen Punktezähler, der für jedes erzeugte Auto hochgezählt wird. Nicht in der Skizze erkennbar ist der zunehmende Schwierigkeitsgrad: Je weiter das Spiel fortschreitet, desto mehr hinderliche Autos tauchen auf. Was Agent X nicht weiß: Er wird nie das Hauptquartier erreichen, das Spiel hat kein Happy End. Es geht einzig und allein darum, wie weit die Spieler es schaffen, bevor ihnen der Strom ausgeht oder sie einen Unfall bauen. In dem Fall erscheint dann ein »Game over«-Schriftzug.

So weit das Spielkonzept. Hast du schon eine Idee, wie das mit Scratch bewerkstelligt werden kann? Denk ruhig einen Moment darüber nach, bevor du weiterliest.

1.2 Auto lenken ohne Führerschein

Eines vorweg: Du bist inzwischen ein echter Scratch-Experte, daher werde ich in diesem Kapitel nicht jedes Puzzleteil einzeln erklären, das du schon kennst. Stattdessen zeige ich dir, welche Figuren du benötigst, und erläutere deren komplette Puzzleskripte. Du findest übrigens alle nötigen Grafiken im Download-Paket des Buches (*www.rheinwerk-verlag.de/spiele-programmieren-mit-godot/*) – oder du zeichnest sie selbst. Beginne also ein leeres Scratch-Projekt, lösche die Katze und du bist bereit!

1 Los geht's mit der Bühne. Klick sie an (rechts unten), wähle die Registerkarte **Hintergrundbilder** und lade die Landschaft mit Straße hoch (**strasse.png**). Beachte, dass der Hintergrund keine gestrichelte Linie in der Straßenmitte enthält. Diese wird später mit einem Trick darauf gemalt und simuliert Bewegung.

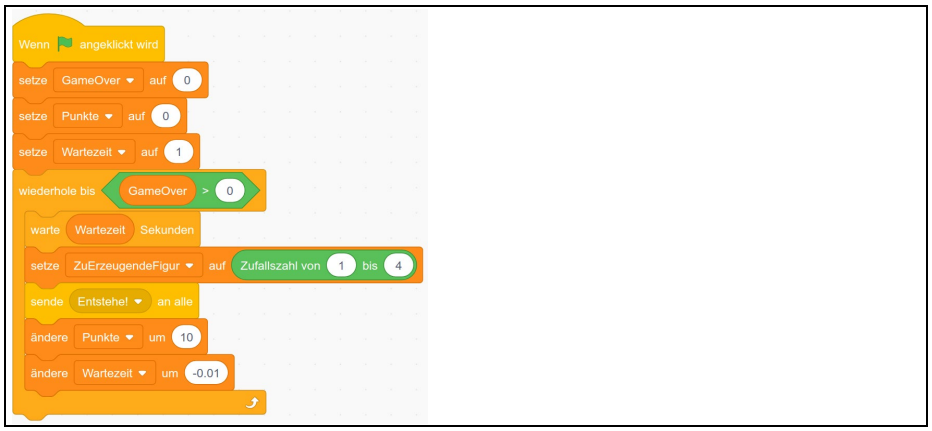
2 Erzeuge folgende Variablen: **Punkte**, **GameOver**, **Wartezeit**, **ZuErzeugendeFigur**. Nur die Variable **Punkte** soll sichtbar sein, bei den anderen entfernst du die Haken:

Die Punktanzeige erscheint oben links auf der Bühne. Du kannst sie mit der rechten Maustaste anklicken und auf **Großanzeige** stellen, das sieht etwas cooler aus.

Die Variable **GameOver** bestimmt, ob das Spiel noch läuft (Wert 0) oder bereits zu Ende ist (Wert 1). Die **Wartezeit** ist die Zeit zwischen dem Erscheinen von zwei Autos. Sie wird immer kürzer, sodass immer mehr Autos im Weg sind. Die letzte Variable, **ZuErzeugendeFigur**, funktioniert genau wie im Luftballonspiel und legt fest, welches der verfügbaren Autos erscheint.

3 Die Bühne ist für den allgemeinen Spielablauf zuständig. Zum Beispiel reagiert sie auf die Nachricht **GameOver**, die bei einer Kollision gesendet wird oder wenn die Batterie leer ist. Füge der Bühne das folgende Mini-Skript hinzu, das in dem Fall den Spielzustand, also die Variable **GameOver**, auf 1 setzt:

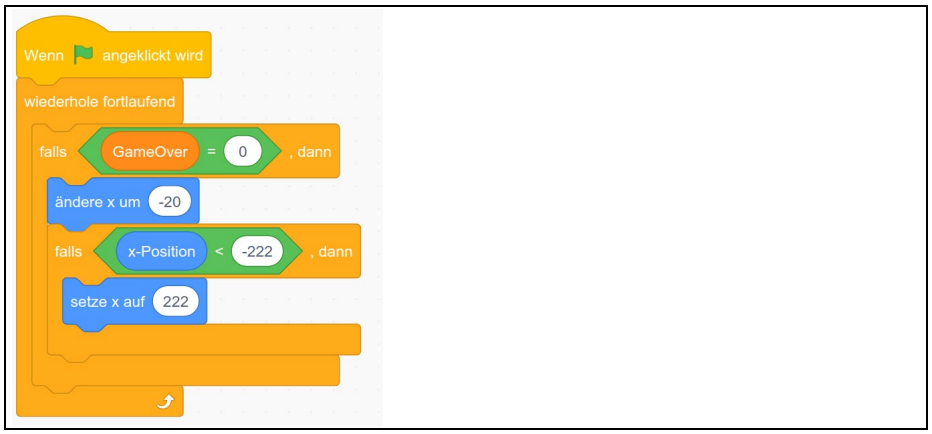
4 Das Spiel wird mit der grünen Fahne gestartet. Füge der Bühne als zweites Skript die Hauptschleife des Spiels hinzu:



Zuerst setzt das Skript den Spielstatus zurück, also **GameOver** auf 0 und die Wartezeit auf 1 Sekunde. Es folgt die Hauptschleife, die wiederholt wird, bis **GameOver** größer als 0 ist.

Erst lässt die Hauptschleife etwas Zeit verstreichen, dann bestimmt der Zufallsgenerator, welche Figur zu erzeugen ist. Diesen Mechanismus kennst du schon vom Luftballonspiel. Die Nachricht **Entstehe!** wird später die Autos und die Nachladebatterien dazu bringen, einen Klon zu erzeugen. Schließlich erhöht die Hauptschleife noch die Punktzahl und verringert die Wartezeit um 0,01 Sekunden. Je größer dieser Wert, desto schneller steigt der Schwierigkeitsgrad des Spiels.

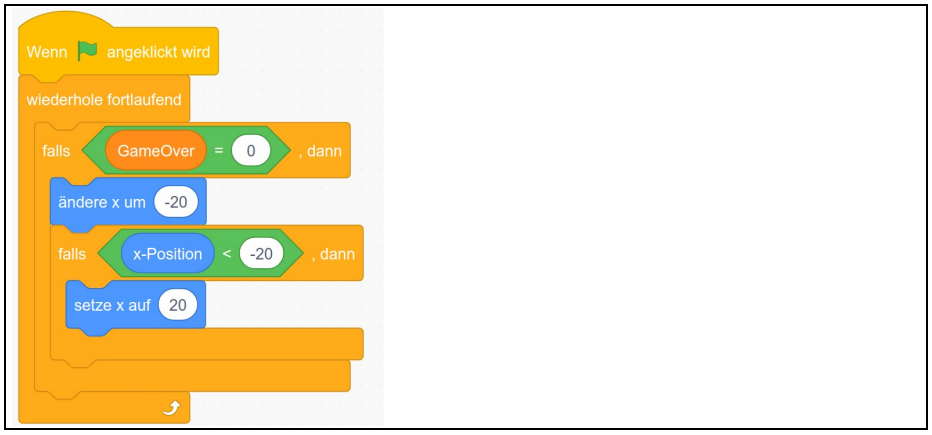
5 Füge vier neue Figuren hinzu und verwende die Grafiken **busch1.png**, **busch2.png**, **busch3.png** und **striche.png**. Positioniere die Büsche auf der Bühne irgendwo am oberen Rand der Straße und die gestrichelte Linie genau in der Mitte. Die Büsche und die Linie sollen sich schnell von rechts nach links bewegen und dann wieder rechts auftauchen, wohlgemerkt nur, wenn das Spiel läuft. Puzzle das zugehörige Skript in der Figur **busch1** zusammen:



Der Busch bewegt sich in jedem Schleifendurchlauf um 20 Einheiten nach links. Das ist ganz schön fix, aber das muss so sein, denn das ist letztlich das wahnsinnige, simulierte Tempo des Sportwagens von Agent X. Sobald der Busch in etwa den linken Rand erreicht hat (x ist kleiner als –222), hüpft er einfach wieder nach rechts.

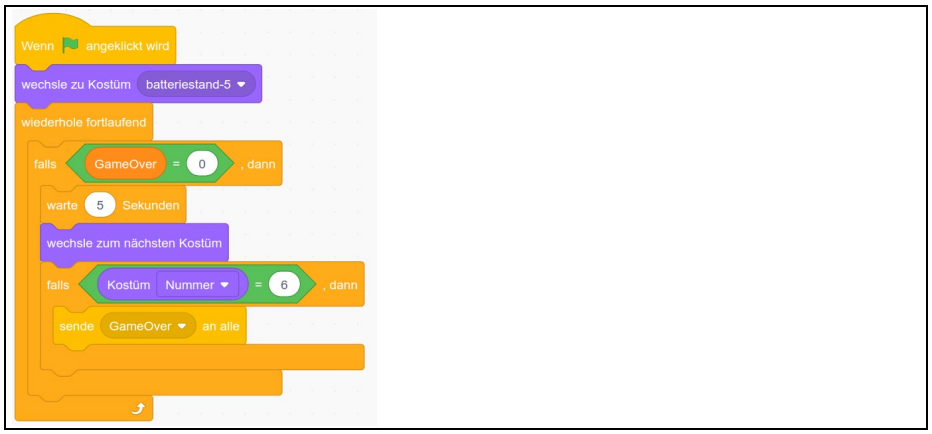
Kopiere dieses Skript in die Figuren **busch2**, **busch3** und **striche**, indem du es mit der Maus jeweils darauf ziehst.

6 Das Skript in der Figur **striche** musst du leicht verändern. Es bewegt sich immer nur zweimal um 20 Einheiten nach links und hüpft dann wieder um die entsprechende Entfernung (2 mal 20, also 40) nach rechts. Das Skript sieht dann so aus:



7 Füge die Batterieanzeige als Figur hinzu. Diese besteht aus sechs Kostümen mit unterschiedlichen Ladezuständen. Sortiere sie so, dass der

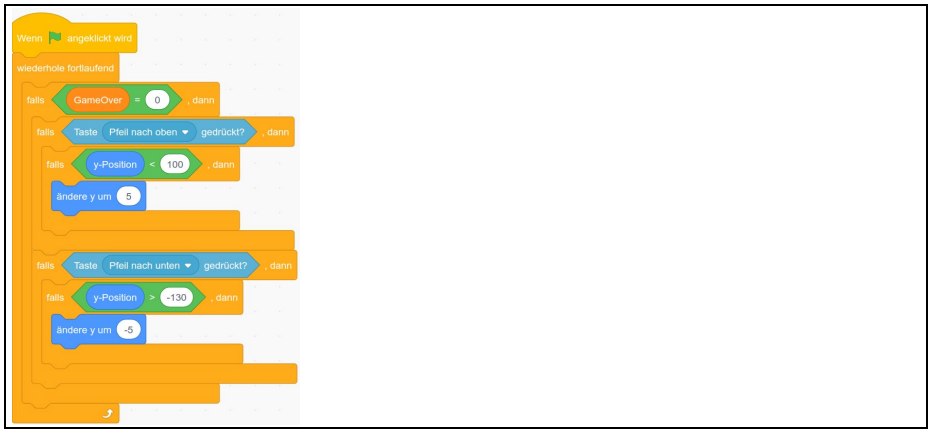
volle Zustand das erste Kostüm ist und die leere Batterie das letzte. Schiebe die Figur nach rechts oben und füge dieses Skript hinzu:



Das Skript setzt beim Start des Spiels den Ladezustand auf voll. Alle 5 Sekunden wechselt es auf den nächstniedrigeren Ladezustand mit dem Puzzleteil **wechsle zum nächsten Kostüm**. Jetzt weißt du auch, warum es auf die Reihenfolge der Kostüme ankommt.

Sobald das Kostüm die Nummer 6 hat (Batterie leer), sendet das Skript die **GameOver**-Nachricht.

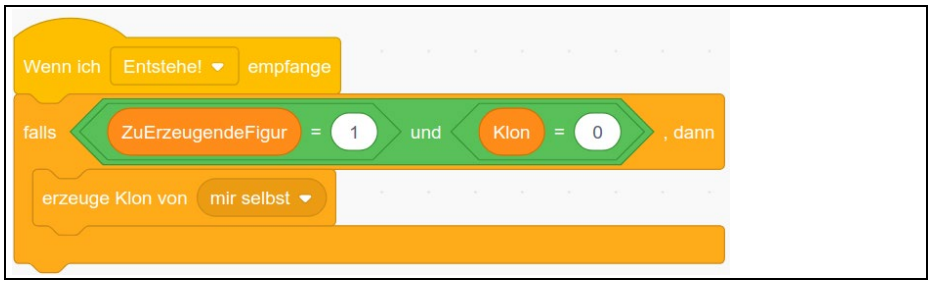
8 Jetzt kommt der Sportwagen von Agent X ins Spiel. Lade **auto-agent.png** als Figur und setze es am linken Bildschirmrand auf die Straße. Puzzle dieses Skript dazu:



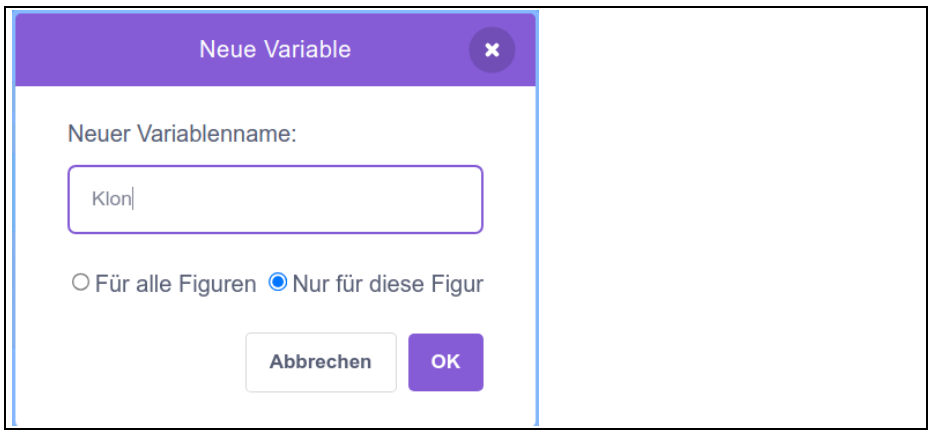
In der Schleife prüft das Skript zunächst, ob das Spiel noch läuft. Nur dann dürfen Spieler den Sportwagen bewegen. Wenn die Pfeiltasten gedrückt werden, ändert das Skript die vertikale Position – aber nur, wenn sie

zwischen –130 und 100 liegt. Sonst könnten Spieler den Sportwagen von der Straße lenken, und das wäre mit Sicherheit eine schlechte Idee.

9 Jetzt sind die Autos an der Reihe. Erzeuge drei neue Figuren **auto1** bis **auto3** aus den drei zugehörigen Autobildern und schalte sie alle drei unsichtbar. Mache dasselbe mit der Batterie (**batterie.png**, nicht zu verwechseln mit dem Batteriestand). Füge zunächst dieses einfache Skript hinzu:

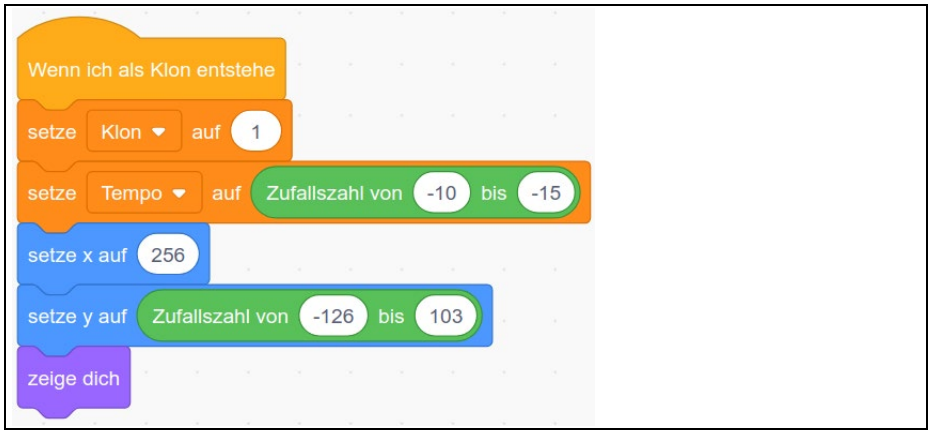


Diesen Mechanismus kennst du schon vom Luftballonspiel. Es gibt nur eine Spezialität: Die Klone dürfen sich nicht vervielfältigen. Da es kein spitz gezacktes Puzzleteil »ich bin ein Klon« gibt, ist eine Variable »Klon« erforderlich. Diese ist zunächst 0 und wird bei allen Klonen auf 1 gesetzt. Wichtig ist, dass diese Variable nur für die jeweilige Figur gelten darf. Das musst du beim Erzeugen der Variablen entsprechend anklicken:



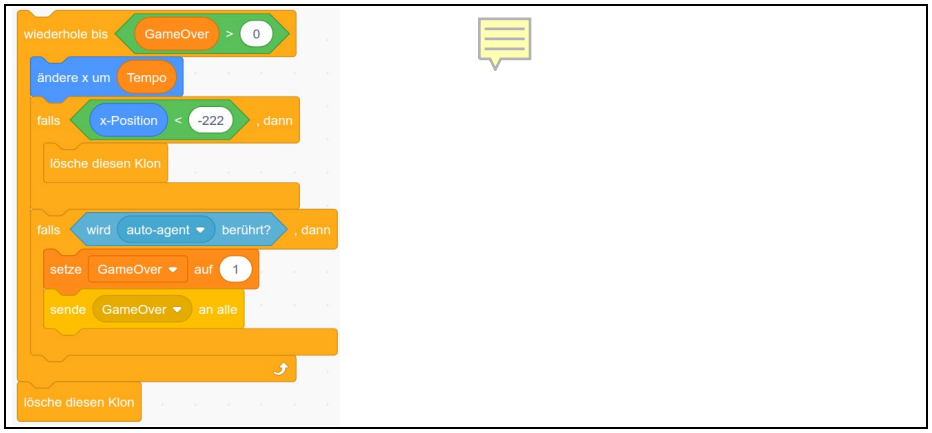
Kopiere das Skript in die beiden anderen Autos und in die Batterie. Ändere jeweils die Zahl, mit der **ZuErzeugendeFigur** verglichen wird, auf 2, 3 und 4.

10 Ein längeres Skript ist fällig für die Entstehung als Klon. Danach muss das Auto sichtbar und bewegt werden. Die obere Hälfte des Skripts muss so aussehen:



Natürlich muss das Skript zunächst »seine« Variable **Klon** auf 1 setzen. Anschließend legt das Skript die Geschwindigkeit dieses Autos fest. Dazu musst du die Variable **Tempo** anlegen – natürlich auch nur für diese Figur. So kann jeder Klon ein anderes Tempo haben, da jeder seine eigene Variable hat. Schließlich wird das Auto auf seine Startposition irgendwo am linken Rand gesetzt und sichtbar gemacht. Beachte, dass die Zufallszahl des Tempos zwischen 10 und 15 liegt – was langsamer ist als das Tempo der Büsche. Das erzeugt die Illusion, dass der Agent schneller fährt als die anderen Autos.

11 Der Rest des Skripts sorgt für die Bewegung des Autos von links nach rechts. Das geht so:



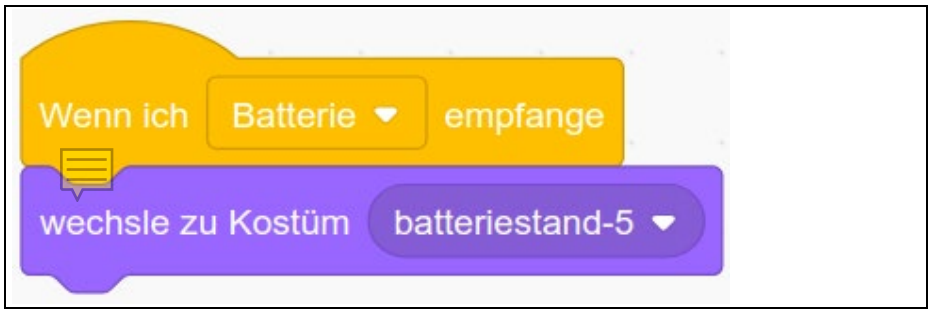
Im Grunde funktioniert die Bewegung wie bei den Büschen, nur langsamer und in entgegengesetzter Richtung. Außerdem wird der Klon einfach gelöscht, wenn das Auto rechts angekommen ist.

Spielentscheidend ist die Berührungsprüfung im unteren Teil. In dem Fall einer Kollision ist das Spiel vorbei, genau wie im Fall einer leeren Batterie.

Kopiere auch dieses Skript in die beiden anderen Autos und in die Batterie.

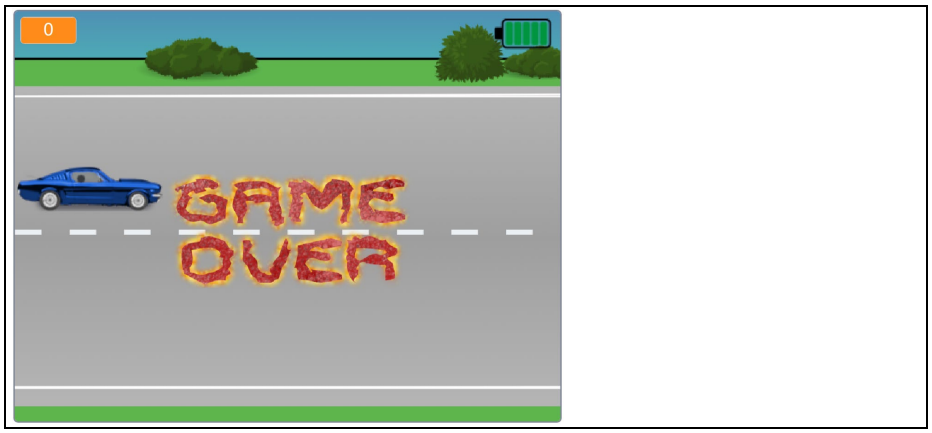
Die Batterie sendet allerdings nicht die Nachricht **GameOver**, sondern die Nachricht **Batterie**. Außerdem muss sich die Batterie genauso langsam bewegen wie die Büsche, also um –20. Um die Batterie »einzusammeln«, füge das Puzzleteil **lösche diesen Klon** hinzu. Notwendig ist das aber nicht.

12 Die Anzeige des Batteriestandes muss sich füllen, wenn der Agent eine Batterie von der Straße aufsammelt. Dafür wird es natürlich eine Nachricht geben, füge also der Batterieanzeigefigur dieses einfache Skript hinzu:



13 Jetzt fehlt nur noch der Schriftzug »Game over«, der am Ende des Spiels erscheint. Auch dies ist eine Figur (**gameover.png**). Lade sie und positioniere den Schriftzug genau in der Mitte des Bildschirms. Sie erhält zwei ganz einfache Skripte, die ich dir gar nicht weiter erklären muss:

Die Bühne sieht jetzt so aus:



Starte das Spiel und schau, ob alles funktioniert. Wenn nicht, überlege, wo der Fehler liegen könnte. Vergleiche noch mal alle Skripte. Manchmal ist es nur eine Kleinigkeit.

Du kannst das Spiel noch weiter ausbauen. Hier ein paar Vorschläge, die teils ganz schön knifflig sind:

- Ändere die vertikale Bewegungsgeschwindigkeit des Agentenautos bei Tastendrücken und teste, ob das Spiel dadurch einfacher oder schwerer wird.
- Füge Geräusche beim Zusammenstoß, beim Aufsammeln einer Batterie und bei Änderungen des Ladezustandes hinzu.
- Wenn der Agent eine Batterie einsammelt, füllt das die Ladung immer komplett auf. Ändere das so, dass die Ladung sich nur um eine Stufe erhöht.
- Füge dem Spiel ein cooles Logo hinzu, das vor dem Start sichtbar ist.

Bevor Agent X irgendwann doch noch sein Hauptquartier erreicht, wird es Zeit, den großen Schritt von Scratch zu Godot zu vollziehen – und zwar schon im nächsten Kapitel!