Leitfaden zum Erstellen eines Frogger-Spiels: Teil 1

Ziel dieser Dokumentation ist es, Sie beim Erstellen eines Frogger-Spiels zu unterstützen. Am Ende dieses Leitfadens haben Sie ein funktionierendes Computerspiel erstellt, welches Abbildung 1 ähneln sollte. Es werden einige Begriffe aus der Computersprache vorkommen, die allerdings erklärt werden, bevor sie im weiteren Kontext wiederverwendet werden.

Inhalt

Beschreibung des Spiels

Das folgende Computerspiel entspricht dem Spieleklassiker “Frogger” aus den 80er Jahren. Das ursprüngliche Frogger war ein Arcade-Spiel aus dem Jahr 1981 mit kurzen Spielzyklen, da es auf Automaten gespielt wurden und diese natürlich möglichst viel Geld damit machen wollten. Es gibt unterschiedliche Schwierigkeitsgrade, wobei wir uns in der folgenden Dokumentation auf einen „Level“ beschränken werden. Das Spiel lässt sich in Anlehnung an das Original wie folgt beschreiben:

Du bist ein Frosch. Deine Aufgabe ist einfach: hüpfe über eine befahrene Strasse, weiche Fahrzeugen aus, bis du ans Ufer eines Flusses gelangst, wo du ohne zu ertrinken zur Grotte ans andere Ufer gelangen sollst, indem du auf den Rücken von Schildkröten und Baumstämme spingst. Hüte dich vor Schlangen und Krokodilen ! (Sega, 1980)

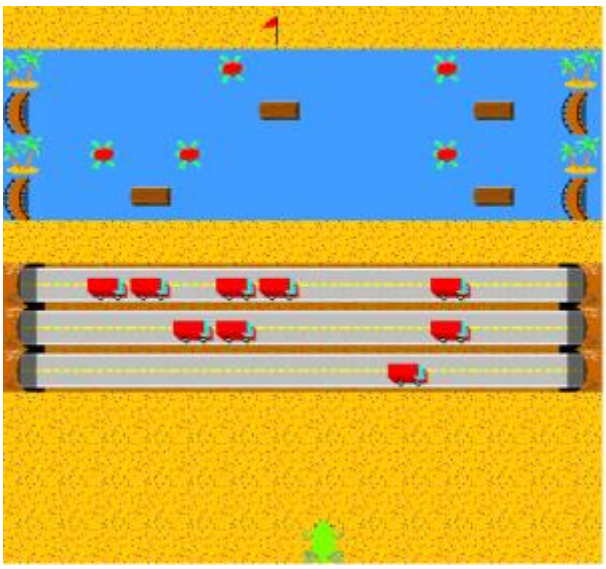


Abbildung 1: Beispiel für Frogger Spiel

Der Frosch beginnt das Spiel am unteren Rand des Bildschirmes (das heißt, dort wurde er initial platziert). Der Spieler kann den Frosch mittels Tastatureingabe in 4 verschiedene Richtungen bewegen (wir haben uns in dieser Dokumentation für die Pfeiltasten entschieden, beliebt ist aber auch die Tastenkombination „W“, „A“, „S“, „D“): rechts, links, oben, unten. Ziel des Spieles ist es, dass der Frosch zur Grotte gelangt. Dazu muss der Frosch auf einer viel befahrenen Strasse Fahrzeugen ausweichen und über einen Fluss mit diversen Hindernissen gelangen.

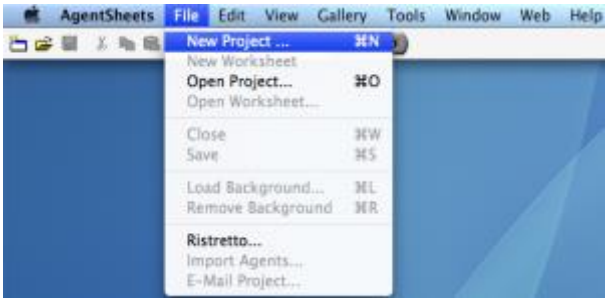
Folgen Sie den Anweisungen dieses Leitfadens möglichst genau und lassen Sie keine Programmierschritte aus. Nach der Einführung ins Programmieren mittels dieses Spiels werden Sie viele Konzepte des Programmierens und des logischen Aufbaus von Computerprogrammen verstanden haben und in der Lage sein, weitere Spiele zu bauen (nach einiger Übung auch Spiele, die Sie oder ihre Schüler sich selbst ausgedacht haben). Viel Erfolg !

Ein neues Projekt entsteht

Abbildung 2: Die Gallerie

Mit einem Doppelklick auf das AgentSheets Zeichen (icon) startet das Programm.

Selektieren Sie File->New Project... um ein neues AgentSheet Projekt zu starten. Danach öffnet sich ein Dialogfenster, in welchem dem Projekt ein Name gegeben werden kann und man definieren kann, wo man das Projekt speichern möchte.



Wir nennen das Projekt "Frogger1" und klicken *OK*. Da es in der AgentSheets Software bereits ein vorgefertigtes Frogger-Spiel mit dem Namen Frogger gibt, sollte der Name für unser Computerspiel ein anderer sein. Z.B. *Frogger1* oder *FroggerPT* (mit den Anfangs-initialen des Namens). Gerade wenn man eine Schulklasse unterrichtet, macht es Sinn, dass der Name des Projektes vielleicht eine Verbindung zum Schüler oder der Klasse herstellt. Bestätigen Sie die Eingabe mit der Aufforderung *OK*.

Nun erscheint ein weiteres Dialogfenster, in welchem die **Pixelgrösse der Objekte** definiert wird. Hier ist per *default* (= Voreinstellung) die Pixelgrösse 32x32 hinterlegt. Diese übernehmen Sie mit *OK*. Nun sollte ein Fenster mit der Galerie (gallery) auf dem Bildschirm links oben erscheinen.

Das Erstellen von Objekten

Im Folgenden erstellen wir graphisch verschiedene Objekte, in der visuellen Programmierung auch “Agenten” (vom englischen Wort *agents*) genannt. Wir beginnen mit dem Frosch.

Wir erstellen einen Frosch

Wenn man im Menüfeld oben auf **“Gallery” klickt**, erhält man eine Auswahl und wählt dort die Möglichkeit **„New Agent“** (neues Objekt). Wir nennen diese Objekt (= *agent*) „Frosch“. Dieser sollte nun in dem neuen Dialogfenster der Galerie auf der linken Seite des Bildschirmes erscheinen (mit dem Bild von Abbildung 3 - als Voreinstellung hinterlegt). Nun gestalten wir den Frosch.

Durch Doppelklick auf das Namensfeld “Frosch” (derzeit noch mit einem Gesicht) oder indem ich auf die Eingabeoption „Edit Depiction“ (editiere das Bild) klicke, öffnet sich ein neues Dialogfenster Graphik Editor, in welchem ich ein Objekt zeichnen und gestalten kann.

Abbildung 3 Graphik Editor

Klicken Sie zuerst auf "clear", damit der Zeicheneditor für eine neue Bildeingabe bereit ist. Der Mann mit der lila Brille ist nur ein "*default*" Bild (default = Voreinstellung) und kann gelöscht werden, so dass ein leeres Graphikfenster neue Zeichnungen erlaubt.

Nun wollen wir einen **Frosch zeichnen**: Wählen Sie eine Farbe aus und klicken Sie auf den "Stift", um mit dem Zeichnen zu beginnen. Wenn man eine Umrandung gezeichnet hat, kann man die Fläche innerhalb dieser Umrandung mit dem sogenannten "Farbeimer" füllen. Der Farbeimer ist das 4. Symbol (= *icon*) auf der linken Menüseite des Editors. Wenn man eine Eingabe rückgängig machen möchte, kann man dies mit dem Eingabefeld "*undo*" machen.

Außerdem hat man die Möglichkeit, Formen zu zeichnen (runde mit dem Kreissymbol oder rechteckige mit dem Rechteck). Es gibt auch den sogenannten "Radiergummi" (das 3. Symbol auf der linken Seite), mit welchem Fehler entfernt werden können. Interessant für ein graphisch zufriedenstellendes Ergebnis ist noch die Eingabe *Color → Mask Color → White* im oberen Menü des Editors. Damit kann man den Hintergrund transparent machen, was im fertigen Spiel eine bessere Graphik ergibt. Außerdem hat man bei "*Color*" die Möglichkeit, auf eine größere Farbpalette zuzugreifen.

Klicken Sie auf die Schaltfläche "Done" im unteren Bereich des Editors, wenn der Frosch fertig gezeichnet ist. Damit ist die erste Graphik gespeichert, kann aber jeder Zeit über "Edit Depiction" in der Galerie verändert und modifiziert werden.

Das Erstellen der Strasse

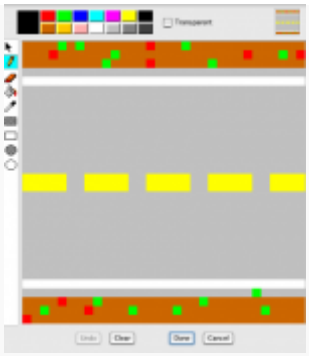


Abbildung 4: Graphische Darstellung des Objektes Strasse

Im Folgenden erstellen wir ein weiteres graphisches Objekt, ähnlich wie wir es zuvor mit dem Frosch gemacht haben: Die Strasse.

In unserem Galerie-Fenster selektieren wir "New Agent", um das Objekt "Strasse" zu gestalten. Wir nennen dieses Objekt “Strasse”. Hier ein **wichtiger Hinweis**: Die Software kennt keine Umlaute und auch kein ß, wie es im Deutschen gebraucht wird. Daher muss man bei der Namensgebung ein wenig aufpassen.

Wir öffnen den Graphik Editor durch Doppelklick auf den Namen des Objekts (also "Strasse") oder durch "Edit Depiction", so wie wir es auch beim Frosch gemacht haben. Die Graphik mit der Strasse sollte in horizontaler Richtung gezeichnet werden, so wie auf der Abbildung zu sehen.

Nun haben wir zwei Objekte (agents): Den Frosch und die Strasse! Prima! Als nächstes bauen wir unser Spiel. Dies geschieht mit dem sogenannten “Worksheet.”

Wir erstellen ein Worksheet

Das Worksheet (Spielfeld) ist unser Spiel-Level bzw. das Feld, auf welchem wir unsere Objekte platzieren.

Abbildung 5: Leeres Spielfeld (worksheet)

Gehen Sie zum Menü in der oberen Leiste des Bildschirms und wählen Sie "*File → New Worksheet*", um ein neues Spielfeld zu erstellen. Es erscheint ein **neues Dialogfenster mit einem leeren Spielfeld** (*Worksheet: Untitled*). Sie können die Größe des Spielfeldes anpassen, indem Sie den Rand des Spielfeldes mit dem Pfeil in die gewünschte Grösse ziehen. Auf dieses leere Spielfeld platzieren wir unsere verschiedenen Objekte (agents), so dass wir am Ende ein fertiges Frogger Spiel (wie in Abbildung 1) bekommen. Wir beginnen mit dem **Frosch und der Strasse** und ergänzen die anderen Objekte nach und nach. Die Strasse verläuft horizontal und um das Spiel interessanter zu machen, platzieren wir mehrere parallele Strassenzüge.

Selektieren Sie auf dem Spielfeldmenü in der linken Menüleiste den Stift, um **Objekte auf das Spielfeld (Worksheet) zu platzieren**. Nun wählen wir in der Galerie das Objekt Strasse aus, indem wir es anklicken. Dabei wird es hellblau umrahmt. Nun können die Strassenelemente einzeln auf dem Spielfeld platziert werden. Schneller geht es, wenn man in der Menüleiste des Spielfeldes das ausgefüllte Rechtecksymbol wählt. Damit kann man mehrere Strassenzüge gleichzeitig zeichnen. Wie auch schon beim Graphikeditor, kann man mit dem Radiergummi eventuelle Fehler löschen. **Wichtig** ist es zu verstehen, dass dieses **Spielfeld in Schichten (Ebenen oder *layer*) aufgebaut** ist. Das heisst, wenn ich an einer Stelle etwas übermale, d.h ich platziere ein Objekt über ein anderes, so kann es sein, dass später beim Programmieren und testen, Dinge passieren, die man sich erst mal nicht erklären kann. Wenn man dann auf dem Worksheet an einer Stelle vielleicht mit dem Radiergummi eine Ebene wegradiert, erkennt man, dass ein Objekt unter einem anderen versteckt war. Frosch und Strasse sollten auf der unteren Hälfte des Spielfeldes positioniert werden, da zu einem späteren Zeitpunkt noch der Fluss ergänzt wird.

Wir positionieren den Frosch auf unserem Spielfeld (Worksheet): Ähnlich, wie gerade mit der Strasse, wollten wir nun einen einzelnen Frosch auf das Worksheet platzieren. Zuerst selektieren wir den Frosch in der Galerie (er bekommt einen hellblauen Rahmen), danach klicken wir auf das Symbol des Stiftes in unserem Worksheet Menü und nun können wir den Frosch unterhalb der Strasse setzen.

Nun ist es ein guter Zeitpunkt, um das Worksheet (Spielfeld) zu speichern !

WICHTIG: Wir speichern das Worksheet

Während das Fenster mit dem Worksheet selektiert ist, gehen wir in der Menüleiste auf ***File → Save***; Geben Sie dem Worksheet einen passenden Namen (z.B. Level 1) und drücken Sie speichern (save). Man kann testen, ob das Worksheet gespeichert wurde, indem man im Worksheet Fenster die Schaltfläche "*reset*" drückt. Wenn alles auf dem Worksheeterhalten bleibt, dann haben Sie korrekt gespeichert. Anderenfalls würde das Worksheet wieder leer sein.

Wir testen das Programm

SUPER! Sie haben nun die erste Version eines Frogger Spiels gebaut. Nun testen wir unser Programm und versuchen, es zu spielen.

Im Worksheet Fenster klicken Sie auf die Schaltfläche "Run". Was passiert ?

- **Bewegt sich der Frosch oder bleibt er wo er ist ?**
- **Können Sie durch Tastatureingabe den Frosch bewegen ?**

Die Antwort lautet nein. Damit der Frosch sich bewegt, müssen wir ihm ein bestimmtes Verhalten (behavior) zuordnen.

Wir programmieren die Bewegungen des Frosches

Nun geht es darum, den Objekten ein bestimmtes Verhalten zuzuordnen. Wie bewegen und interagieren die Objekte während des Spiels ? Wir fangen mit dem Frosch an. Er soll sich in verschiedene Richtungen bewegen, wenn wir bestimmte Tasten auf dem Keyboard bedienen.

Abbildung 6: Der Behavior Editor (das ist der Editor für die Programmierbefehle des Objektverhaltens) mit einer Spalte für "If" Eingaben und einer Spalte für "Then" Eingaben (wenn-dann Befehle).

Über den "*Edit Behavior*" Befehl öffnen wir den behavior editor des Frosches. Im Galerie Fenster selektieren Sie das Frosch Objekt und drücken dann auf "Edit Behavior" unten im Galeriefenster. Dadurch wird ein neues Dialogfenster geöffnet (Behavior Editor), welcher noch ohne Befehle ist. Wir nennen die Befehle, welche wir dem Programm zuordnen (programmieren) Regeln. Regeln bestehen aus einer **WENN-DANN (if-then) Bedingung**. Wenn wir den Frosch mit den Pfeiltasten bewegen wollen, dann lautet eine Regel: "Wenn die Pfeiltaste ↑ gedrückt wird, dann soll sich der Frosch um einen Schritt nach oben bewegen". Insgesamt brauchen wir 4 Regeln, welche die Bewegungsrichtungen anordnen (oben, unten, rechts, links).

Mit einem **Doppelklick auf die "If" Box**, bekommen wir die Übersicht der Befehle für Bedingungen (conditions). Alternativ können Sie über das Menü gehen und dort bei "Tools → Conditions Palette" das Fenster mit den Bedingungen öffnen (blau hinterlegte Befehle).

Tastatur-Eingabe

Mit einem **Doppelklick auf die "Then" Box**, bekommen wir die Übersicht der Befehle für die Aktionen (actions). Alternativ können Sie über das Menü gehen und dort bei "Tools → Actions Palette" das Fenster mit den Aktionen öffnen (rot hinterlegte Befehle).

Wir programmieren die "Tastatur-Eingabe" (key condition) für den Frosch. Klicken Sie auf Symbol *Key A* im Auswahlfenster der Konditionen (conditions) und ziehen Sie diesen Programmierbaustein mit der Maus (*drag & drop*) in den Programmiereditor (*behavior Frosch*) in die linke Spalte mit der *if* Bedingung. Aus dem "*A*" können wir durch einen Klick auf das Buchstabensymbol eine beliebige Tastatureingabe machen und wählen in diesem Fall die **Pfeiltaste** →. Die Regeln bestehen aus zwei Teilen, es fehlt nun noch die *then* (dann) Bedingung.

Eine Bewegung nach rechts

Wenn die Pfeiltaste → vom Spieler gedrückt wird, soll sich der Frosch mit einer Bewegungseinheit nach rechts bewegen.

Wir ergänzen die Bewegung in den Programmierteil für "Aktionen" (*actions*). Dazu ziehen Sie den Programmierbaustein *Move* aus dem Dialogfenster der *actions* in die rechte Spalte des Programmiereditors für den Frosch, welcher die Bezeichnung *then* (dann) trägt. Durch einen Klick auf den Pfeil im Programmierbaustein *Move* kann die Bewegungsrichtung angepasst werden. Selektieren Sie den Pfeil passend zu ihrer Tastatureingabe. Nun sollte der Programmiereditor (*behavior editor*) des Frosches wie folgt aussehen.

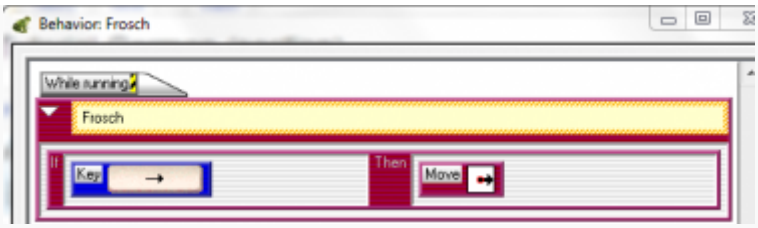


Abbildung 7: Der Frosch bewegt sich nach rechts: "Wenn die Pfeiltaste → bedient wird, dann bewege dich nach rechts"

Klicken Sie auf **"*Apply*"** (entspricht speichern)im unteren Bereich das Programmiereditors, um die Eingaben zu speichern.

Programmier-Test: Bewegt sich der Frosch ?

Nun wollen wir das Programm gleich testen, indem wir es starten. Sie klicken auf den *grünen Pfeil* in ihrem Worksheet-Dialogfenster, welcher das Computerspiel initiiert. Was passiert ?

- Der Frosch sollte sich nach rechts bewegen, wenn Sie die rechte Pfeiltaste auf ihrer Tastatur bedienen. Sollte sich der Frosch nicht bewegen, schauen Sie in der Dokumentation nach, woran es liegen könnte.
- Wenn sich der Frosch korrekt nach rechts bewegt, sind wir dann schon zufrieden mit den Bewegungsmöglichkeiten für den Frosch ? Welche anderen Bewegungen brauchen wir für unser Frogger-Spiel ?

Nach erfolgreichem Testdurchlauf gehen wir auf ***reset***, um den Frosch in die Ausgangslage zurückzubringen. Durch "Reset" wird das Spiel immer wieder auf die gespeicherte Version zurückgebracht.

Duplizieren einer Regel: Da wir bislang nur die Bewegung nach rechts programmiert haben, wollen wir nun die restlichen Bewegungsrichtungen ergänzen. Dazu fügen wir 3 weitere Regeln hinzu. Dies können wir durch Kopieren der ersten Regeln mit geringem Aufwand tun, indem wir die bereits vorhandene Regel selektieren (sie sollte dann blau umrahmt sein) und dann klicken wir auf die Schaltfläche ***duplicate*** und wiederholen diesen Vorgang drei Mal.

Anschließend müssen wir die neuen Regeln anpassen, indem die Richtungen der Tastatureingaben (also Pfeil mit Richtung nach oben, unten und links) verändert werden und dann auf der Seite der Aktionen die Pfeile der Bewegung ebenfalls passend angepasst werden. Schliesslich sollte der Programmiereditor (*behavior editor*) wie folgt aussehen.

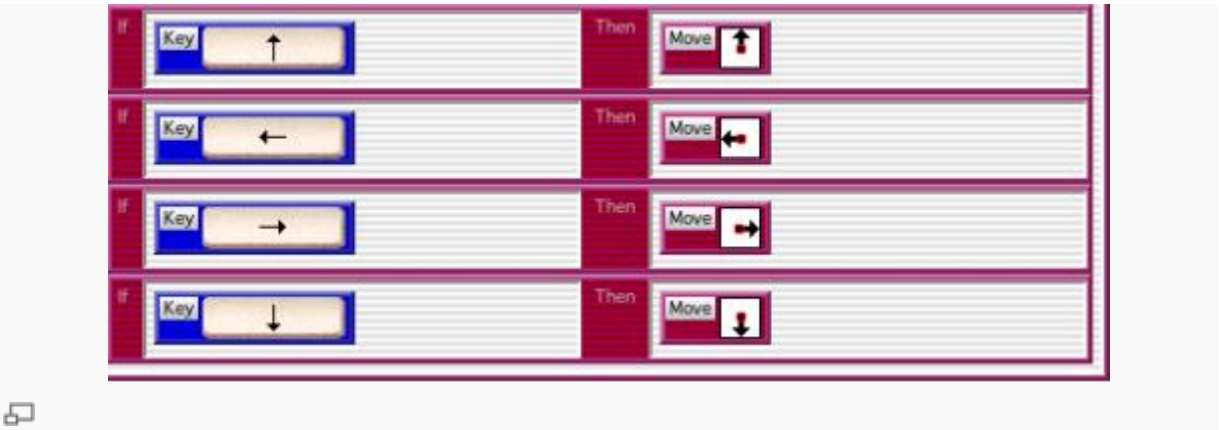


Abbildung 8: Fertiger Programmausschnitt für alle vier Bewegungsrichtungen des Frosches

Run

Programm-Test: Wir testen die Bewegungsabläufe des Frosches

Testen Sie das Programm, indem Sie den Frosch in alle vier Richtungen hüpfen lassen.

- Sollte es nicht funktionieren, vergleichen Sie ihr Programm nochmal mit den Abbildungen der Dokumentation.
- Wenn es funktioniert - SUPER - WEITER SO!

Wir erstellen und programmieren ein Fahrzeug

Unser Spiel hat jetzt einen Frosch, der sich nach oben, unten, rechts und links bewegen kann. Aus den vorausgegangenen Schritten haben wir gelernt, wie man ein Objekt (agent)

erstellt und diesem grundlegende Verhaltensanweisungen zuweist. Nun wollen wir ein Fahrzeug erstellen, welches auf sich auf der Strasse bewegt.

Fahrzeug Objekt (agent)

Wir erstellen und zeichnen das Fahrzeug Objekt. Analog zu unserem Frosch und der Strasse, erstellen wir ein neues Objekt (agent) und nennen dieses „Fahrzeug“ (oder Auto).

Anschließend programmieren wir die Bewegungen des Fahrzeugs. Das Fahrzeug soll sich entlang der Strasse von links nach rechts bewegen. Dazu öffnen wir zuerst den Programmiereditor (*behavior-editor*). Das Fahrzeug soll sich in bestimmten Zeitabständen entlang der Strasse bewegen (nach rechts), das heißt, Bedingung für die Bewegung ist, dass das Fahrzeug rechts von sich die Strasse sieht und sich die Bewegung in regelmäßigen Zeitabständen wiederholt. Dazu ziehen wir mit der Maus (*drag & drop*) den Programmierbaustein „see“ und klicken auf das Bild mit dem roten Punkt, welches uns die Option zur Richtungseingabe gibt. Wir wählen den passenden Pfeil (→) und ändern das Symbol für unsere Objektauswahl *(agents)* in das Symbol mit der Strasse. Danach fügen wir noch den Programmierbaustein „*Once every … Secs*“ hinzu und geben als Zeiteinheit 0.5 Sekunden ein. Bei den Aktionen (das sind die roten Programmierbausteine) ziehen wir den Befehl „*move*“ in die Regel auf der rechten Seite bei den Aktionen (*actions*). Zum Schluss ändern wir hier wieder die Richtungseingabe auf einen Pfeil in Fahrtrichtung (also nach rechts). Das Programm für unser Fahrzeug sollte nun wie folgt aussehen:

Abbildung 9: Programm für das Verhalten der Fahrzeuge: Das Fahrzeug bewegt sich entlang der Strasse alle 0.5 Sekunden ein Feld weiter.

Wir testen das Bewegungsverhalten des Fahrzeugs

Wir testen unser Programm, um zu prüfen, ob sich das Fahrzeug-Objekt richtig bewegt. Wir zeichnen in unserem Spielfeld-Dialogfenster (*Worksheet window*) einige Fahrzeugobjekte auf die Strasse, speichern das Worksheet und drücken den „*Run Knopf*“.

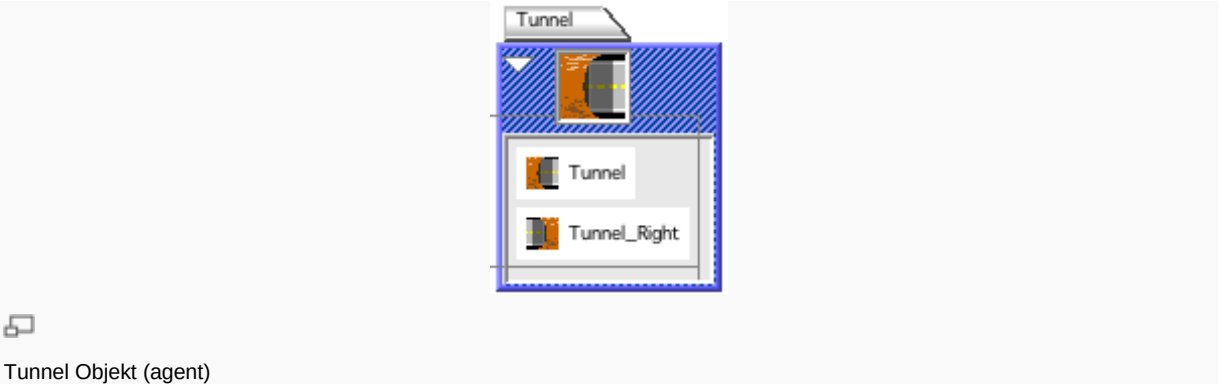
- Bewegen sich die Fahrzeuge ? Sollten sie sich nicht bewegen, kontrollieren Sie bitte ihr Programm im Fahrzeugobjekt. Stellen Sie sicher, dass rechts von ihren Fahrzeugen ein Stück Strasse liegt. Wenn sich die Fahrzeuge bewegen, lassen sie das Programm einen Moment laufen.
- Was passiert, wenn die Fahrzeuge das Ende der Strasse erreichen ? Verschwinden die Fahrzeuge oder gibt es einen Stau ?
- Wie können weitere Fahrzeuge von links nachkommen, wenn alle das Ende der Strasse erreicht haben ?

Wir brauchen also Fahrzeuge, die auf der linken Seite des Spielfelds auf der Strasse entstehen, dann die Strasse entlang nach rechts fahren und am rechten Ende wieder verschwinden. Bislang fahren sie einfach nur.

Wir benötigen ein „Hilfsobjekt“, welches für das Entstehen und Verschwinden der Fahrzeuge zuständig ist. Dazu wählen wir verschiedene Tunnel. Ein Tunnel auf der linken Seite des Spielfeldes, welcher Fahrzeuge entstehen lässt (er spuckt sie sozusagen aus) und ein andere Tunnel auf am rechten Ende der Strasse, welcher die Fahrzeuge verschwinden lässt (dieser verschluckt die Fahrzeuge wieder).

Wir erstellen und programmieren ein Tunnel Objekt

Zuerst zeichnen wir das Tunnel Objekt (agent). Da wir zwei verschiedenen Tunneltypen benötigen (einen, der die Fahrzeuge am Anfang der Strasse generiert und einen weiteren Tunnel am Ende der Strasse, der die Fahrzeuge verschwinden lässt), zeichnen wir in unser Tunnelobjekt (also in ein und denselben agent) zwei verschiedene Tunnelbilder. Man kann also in ein Objekt mehrere verschiedene Objektbilder hinzufügen, welche unterschiedliche Ausrichtungen des Objektes widerspiegeln.



Tunnel Objekt (agent)

Mehrere Bilder in einem Objekt (agent): Wenn wir 2 unterschiedliche Tunnelarten und Bilder benötigen, klicken wir auf das Tunnelobjekt (um es somit für weitere Eingabeoptionen zu selektieren) und dann wählen wir *„New Depiction“* als Eingabe. Dadurch wurde unser erster Tunnel kopiert.

Nun gibt es eine Reihe von graphischen Hilfsmitteln, mit denen wir ein neues Objekt direkt spiegeln oder rotieren können und somit schneller zum Ziel kommen, als wenn wir es mit Stift und Farbeimer neu zeichnen würden. Dazu selektieren wir mit der Maus den Namen des ersten Objektbildes (in diesem Fall *„Tunnel“*) und wählen über die rechte Maustaste ***„Duplicate Depiction“*** eine Kopierfunktion mit Spiegelung oder Rotation in verschiedene Richtungen. Probieren Sie es einfach aus. Über *Duplicate Depiction → Flip Horizontal* bekommen Sie einen zweiten Tunnel, der an das Strassenende passt.

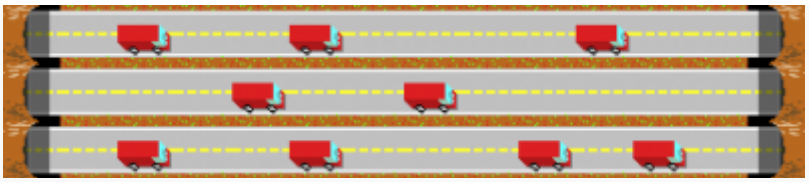


Abbildung 10: Tunnel und Strasse in mehreren Ebenen auf unserem Spielfeld (worksheet)

Nun müssen wir die Tunnel programmieren, das heißt, ihnen ein bestimmtes Verhalten zuordnen. Der linke Tunnel soll periodisch Fahrzeuge generieren. Dazu öffnen wir den Programmeditor des Tunnels (behavior editor) und fügen die Programmierbausteine wie im Bild unten hinzu.

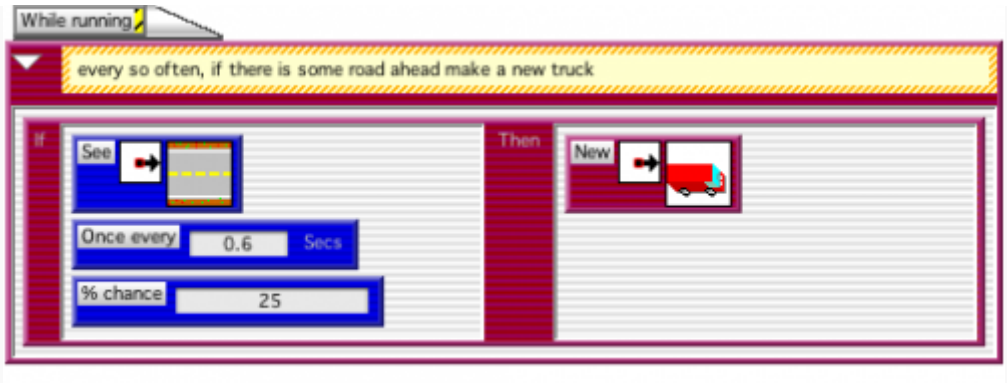


Abbildung 11: Tunnel Programm

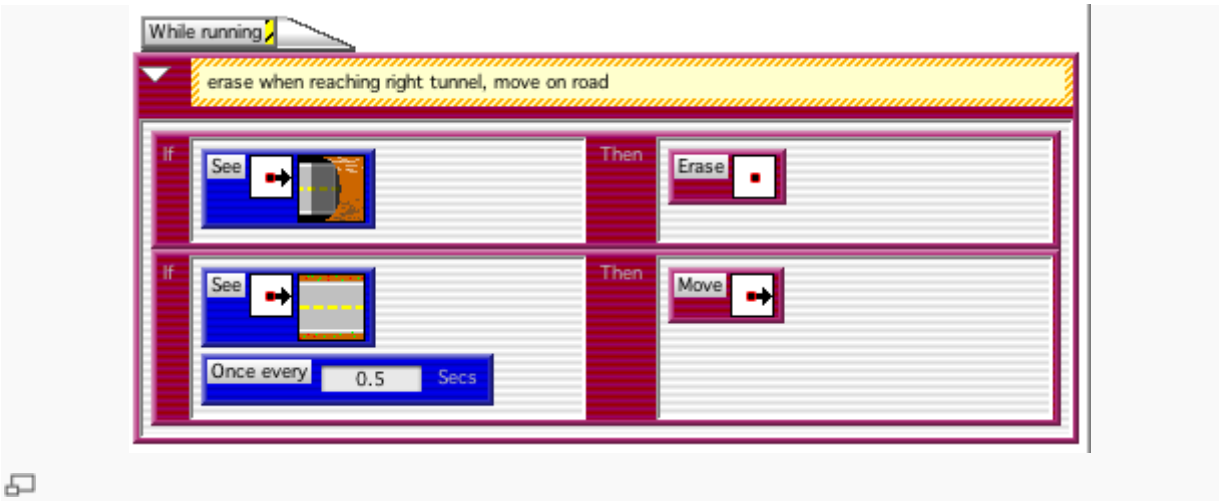
Erklärung: Der Tunnel soll Fahrzeuge generieren, aber nur, wenn rechts von ihm eine Strasse ist. Würde z.B. gerade ein anderes Fahrzeug rechts vom Tunnel stehen, so soll kein neues Fahrzeug generiert werden, da sie sich ja sonst stapeln würden. Um den Spielverlauf

ein bisschen anspruchsvoller zu gestalten, ergänzen wir zur Zeiteinheit noch eine Wahrscheinlichkeit. Dadurch werden die Fahrzeuge in unregelmäßigen Abständen aus dem Tunnel „ausgespuckt“, nämlich in dem Fall oben **mit einer Wahrscheinlichkeit von 25% in einem Zeitabstand von 0.6 Sekunden**.

Wie verschwinden die Fahrzeuge am Ende der Strasse ?

Wir haben jeweils einen Tunnel an das Ende der Strasse platziert. Nun müssen wir den Fahrzeugen mitteilen, dass sie sich löschen, wenn sie den Tunnel vor sich sehen.

Das Fahrzeug soll sich selbst löschen: Wie öffnen das Programmierfenster des Fahrzeugs und ergänzen eine neue Regel "*new rule*". In diese neue Regel ziehen wir über *drag & drop* Bedingungen (conditions) und Aktionen (actions), so dass sich das Fahrzeug selbst löscht, wenn es vor dem Tunnel am rechten Ende der Strasse ist. Das vollständige Programm des Fahrzeug-Objekts sieht dann wie folgt aus:



Vollständiges Programm des Fahrzeugs - Fahrzeug löst sich auf, wenn es vor dem Tunnel ist und bewegt sich, solange es eine Strasse vor sich sieht.

Erklärung: Wenn rechts vom Fahrzeug ein Tunnel steht, soll das Fahrzeug verschwinden, indem es sich selbst löscht. (HINWEIS: der "Punkt" im Richtungs-Parameter bedeutet, dass das Objekt selbst referenziert wird). Wenn rechts vom Fahrzeug ein Stück Strasse liegt, bewegt sich das Fahrzeug alle 0,5 Sekunden ein Stück weiter nach rechts.

Wir testen das Fahrzeug / Tunnel Programm

Nun testen wir unser Programm und schauen, wie unsere Fahrzeuge und Tunnel Objekte interagieren. Bedienen Sie im Worksheet Fenster den grünen Pfeil, den "Run bzw. Start"-Pfeil.

- Entstehen durch die Tunnel am linken Strassenrand neue Fahrzeuge ?
- Werden die Fahrzeuge gelöscht, wenn sie den rechten Tunnel erreichen ?

Wenn eine der beiden Forderungen nicht funktioniert, sollten Sie Ihr Programm nochmal untersuchen und mit der Dokumentation vergleichen. Vielleicht haben Sie auch nur die Tunnel-Bilder / Objekte verwechselt und an der falschen Stelle der Strasse platziert. Wenn es funktioniert, dann können Sie ein wenig experimentieren mit dem Frosch. Was passiert, wenn er seinen Weg mit einem Fahrzeug kreuzt ? Wird der Frosch überfahren ?