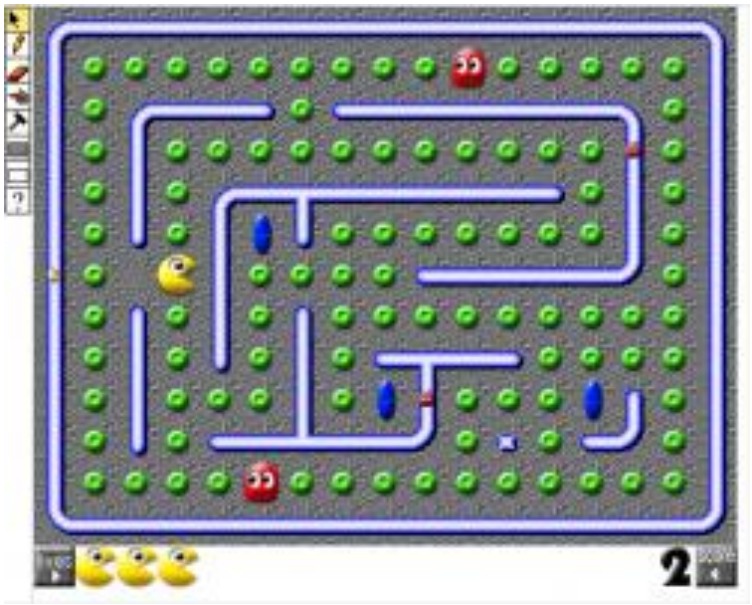
DOKUMENTATION PACMAN



Erstelle das bekannteste Arcade-Spiel der 80er!

Finde den Weg durch das Labyrinth und schlucke dabei Kraftpillen.

Achte auf die Geister!

Ziele:

- Die Schülerinnen und Schüler erstellen eine Version des 80er-Jahre Arcade-Spiels PacMan und benutzen dabei folgende Computational Thinking Muster:
  Benutzer Kontrolle, Absobieren, Kollision, Diffusion, Zählen.

Vorkenntnisse:

- Es wird vorausgesetzt, dass Lernende folgende Fähigkeiten besitzen:
  - Erstellen von Objekten
  - Grundkenntnisse im Bereich Objekt-Verhalten
  - Erstellen eines Frogger Spieles.

Computational Thinking Muster:

- **Benutzer Kontrolle**
- **Kollision**
- **Absorbieren**
- **Diffusion**
- **Zählen**

Länge der Aktivität:

- Drei bis fünf 45 Minuten Lektionen, wobei es sein kann, dass das Tempo je nach Alter der Teilnehmer und Klassengrösse angepasst werden muss.

Beschreibung der Aktivität:

- Teil 1: Erstelle ein Grundspiel mit PacMan und Geistern, die sich zufällig bewegen.
- Teil 2: Veranlasse die Geister dazu, PacMan zu verfolgen.
- Teil 3: Erweitere das Spiel so, dass das Spiel zu Ende ist, sobald die Kraftpillen alle geschluckt sind.

Inhaltsverzeichnis

Vokabeln und Definitionen

Algorithmus	
Attribut	Lokale Variable, die einem Objekt zugeschrieben ist (wie zum Beispiel Geruch).
Klammern	Schreibweise, um Informationen voneinander zu trennen. Hierzu werden eckige Klammern [ ] oder runde Klammern ( ) eingesetzt.
Broadcast	Broadcast bedeutet, dass Daten/Informationen übermittelt werden. Controller (Steuerungseinheit) überträgt (oder sendet) ein Signal/Information oder Unterprogramm.
Geist	Objekt, welches PacMan verfolgt.
Kollision	Ereignis, bei dem zwei Objekte zusammenstossen (Computational Thinking Muster).
Diffusion	Physikalisches Phänomen, bei welchem durch permanente Vermischung eine gleichmässige Verteilung von Teilchen entsteht.
Hill Climbing	Eine spezielle Suchtechnik mathematische Optimierungsmethode / Algorithmus), um eine "lokale" Suche durchzuführen. Es ist ein iterativer Algorithmus, der in unserem Kontext einen lokal grössten Wert ermittelt. Bildlich gesehen ist man ein Bergsteiger in einem Gebirge und sucht im Nebel den höchsten Berg.
Methode	Eine Reihe von Regeln, die in einer bestimmten Situation befolgt werden.
Umfrage	Im Englischen wird der Begriff "polling" benutzt, der auch beim Ermitteln von Wahlergebnissen eingesetzt wird.
Random	In unserem Kontext geht es um eine zufallsgesteuerte Bewegung
PacMan	Hauptcharakter, der die Kraftpillen schluckt, wenn der Spieler ihn auf dem Worksheet hin und her bewegt.

Allgemeine Unterrichtsstrategien[1]

Grundlegende Philosophie

- Bildungsziel ist das Erlernen und Anwenden von *Computational Thinking Mustern* im Kontext eines altbekannten Spiels. Ein Verständnis der Computational Thinking Muster erleichtert Lernenden den Transfer der Programmierkonzepte von bekannten Spielen auf selbst programmierbare Spiele oder Simulationen.
- Direkte Instruktion wird lediglich für jene Inhalte eingesetzt, die die Schülerinnen und Schüler zum ersten Mal erlernen; grundsätzlich wird bei der Dokumentation darauf geachtet, dass Lernende die Programmier-Konzepte verstehen und nicht nur Codes kopieren.
- Die Schülerinnen und Schüler sollen so selbstständig wie möglich arbeiten, mit der notwendigen Unterstützung, wenn sie alleine nicht weiterkommen oder neue Konzepte erarbeiten. Diese Strategie steigert in den meisten Fällen die Motivation der Lernenden.

Unterstützter Entdeckungsprozess

- Modellieren Sie den Prozess, anstatt den Lernenden lediglich die Antwort vorzugeben. Bevor Sie ein Spiel mit Ihren Schülerinnen und Schülern erstellen, sollten Sie das Spiel für sich persönlich erstellen, um mögliche Schwachpunkte oder Stolperstellen zu identifizieren.

[1] Diese Information basiert auf Forschungsergebnissen folgender Publikationen:

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010, June). Using scalable game design to teach computer science from middle school to graduate school. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 224-228). ACM.

National Research Council. (2011). *Learning science through computer games and simulations*. (M. Hilton & M. Honey, Eds.). Washington, DC: The National Academies Press.

National Research Council. (2014). *STEM Integration in K-12 Education:: Status, Prospects, and an Agenda for Research*. (M. Honey, G. Pearson, & H. Schweingruber, Eds.). Washington, DC: The National Academies Press.

Repenning, A., & Ioannidou, A. (2008, March). Broadening participation through scalable game design. In *ACM SIGCSE Bulletin* (Vol. 40, No. 1, pp. 305-309). ACM.

- Lassen Sie die Schülerinnen und Schüler Probleme eigenständig bearbeiten. Stellen Sie Leitfragen oder machen Sie hilfreiche Vorschläge. **Leisten Sie lediglich minimale Hilfestellung**, damit die Lernenden mögliche kleinere Hindernisse selbst überwinden können.

- Unterstützen und fördern Sie **Gruppenarbeit**! Es ist zielführend beim Programmierern, Probleme miteinander zu besprechen und Kodeteile von anderen Programmen oder Programmierern zu verwenden. Das Durchsprechen von Kodierungsproblemen befähigt Lernende, kritischer über *Computational Thinking Muster* sowie die zur Lösung eines Problems notwendigen Schritte nachzudenken. Zu sehen, wie andere ein Problem gelöst haben, verdeutlicht den Lernenden, dass es a) mehrere Lösungsstrategien geben kann und dass b) manche Strategien effektiver als andere sind.

- Erkennen Sie, dass Programmieren ein Prozess des Ausprobierens (**Trial-and-Error**) ist, insbesondere wenn etwas zum ersten Mal gelernt wird. Es ist hilfreich, diese Arbeitsweise den Schülerinnen- und Schülern immer wieder vor Augen zu führen.

Modularer Aufbau

- Das PacMan Spiel hier ist modular aufgebaut, das heisst, Inhalte, welche die Lernenden bereits kennen, werden vorausgesetzt und neue Inhalte werden ausführlich besprochen.

- Stellen Sie sicher, dass Sie die einzelnen Module (bei PacMan insbesondere in den Bereichen Diffusion und Hill Climbing) ausreichend besprechen (diese Computational Thinking Muster werden häufig in künftigen Spielen und Simulationen verwendet).

- Denken Sie daran, dass konzeptuelles Verständnis Zeit braucht. Es kann also sein, dass sie manche Konzepte mehrere Male unter Zuhilfenahme verschiedener Beispiele in unterschiedlichen Situationen erklären müssen, sodass alle Schülerinnen und Schüler erfolgreich sein können.

Unterstützung des Lernens

- Untersuchungen haben ergeben, dass *Scalable Game Design* die meisten Schülerinnen und Schüler extrem motiviert, aber auch konzeptuelles Verständnis schult. Eigenverantwortung und gewisse Freiheiten in der Gestaltung (Eigenverantwortung) steigern die Motivation zusätzlich.

- Programmieren kann für einige Schülerinnen und Schüler schwierig sein und alle Lernenden sind wahrscheinlich frustriert, wenn die Programmierung nicht die gewünschten Resultate zeigt. Loben Sie die Schülerinnen und Schüler, wenn sie kontinuierlich am Problemlösungsprozess arbeiten. Ermutigen Sie die Lernenden, ihre Ergebnisse und Lösungsvorschläge mit anderen zu teilen.

- Vermitteln sie Ihrer Lerngruppe: **Der Prozess ist wichtiger als die Antwort!** Verdeutlichen Sie, dass das Programmieren eines Projektes Zeit braucht. Es ist wichtig, dass Sie den Schülerinnen und Schülern ausreichend Zeit zur Bearbeitung der Aufgaben geben. Üben Sie keinen Druck auf die Lerngruppe aus durch Äusserungen wie „Jetzt aber schnell…“ und verzichten darauf, aus Zeitgründen den Code an die Lerngruppe herauszugeben. Der Prozess des eigenen Aushandelns wird das konzeptuelle Verstehen stärken.

3 Module für PACMAN

Teil (Modul)1:

- Variante 1: Bilden sie heterogene Lerngruppen (Lernende mit Schwierigkeiten in einer Gruppe mit erfahrenden Lernenden)
- Variante 2: Stellen Sie Lernenden mit Schwierigkeiten Handout 1A zur Verfügung. Dieses beinhaltet mehr direkte Instruktionen.
- Zeitmanagement: Das Zeichnen der Objekte (agents) beschäftigt die Lernenden unterschiedlich lang. Es gibt Möglichkeiten, dies ein wenig zu steuern, indem Schüler einige Objekte zu einem späteren Zeitpunkt fertig zeichnen, weniger Design-begeisterte Teilnehmer können auch mal ein Objekt durch "import" hochladen oder aber mit Handout 1B eine Zeichenhilfe zur Hand nehmen..

Teil (Modul) 2:

- Durch Teams mit erfahrenen und unerfahrenen Schülern können viele Probleme einfacher gelöst werden (Mentoren).

Teil (Modul) 3:

- Beachten Sie, dass diese Lektion nicht trivial ist – planen Sie einen höheren Zeitaufwand und beziehen Sie erfahrene Schüler/-innen (Mentoren) ein.
- Mentoren (d.h., erfahrene Schülerinnen und Schüler) können Sie gut unterstützen - vor allem bei der Fehlersuche oder Verständnisproblemen.

Anleitung für die Lehrperson

Teil (Modul) 1

Unterrichtsvorschläge

Geben Sie ihren Schülerinnen und Schülern den Arbeitsauftrag, ein neues Spiel zu programmieren. Die Hauptmerkmale des Spiels sollen folgende sein:

Der Spieler navigiert PacMan durch ein Labyrinth und schluckt dabei Kraftpillen. Wenn alle Kraftpillen geschluckt wurden, erreicht PacMan den nächsten Level. Geister streifen durch das Labyrinth und versuchen, PacMan zu fangen. Sollte PacMan von diesen berührt werden, stirbt er und das Spiel ist vorbei.

Die Lernenden sollen frei über die mögliche Struktur des Spiels (Objekte, Bedingungen und Befehle, Zusammenhänge, künstliche Intelligenz versus Benutzer-gesteuert) diskutieren.

Mögliche Fragen an die Klasse:

- In wie weit ähnelt das Spiel dem Frogger-Spiel? Worin liegen Unterschiede?
- Welche beim Frogger erlernten Programmierschritte können hier verwendet werden?
- Welche Objekte werden benötigt?
- Wie könnte man die Geister programmieren, damit sie sich frei und zufällig bewegen (und was bedeutet "zufällig" in unserem Zusammenhang)?
- Was würde das Spiel spannender machen?

Nachdem die Schüler/-innen über diese Fragen einige Minuten nachgedacht haben, händigen Sie das Handout 1 aus (oder präsentieren es).

Sollten Sie Schüler/-innen haben, die noch nicht mit AgentSheets gearbeitet haben, händigen Sie diesen Handout 1A aus.

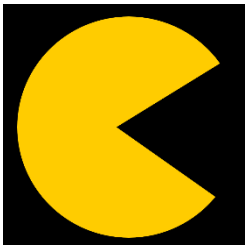
Anleitung für Schülerinnen/Schüler

Teil (Modul) 1 / Handout A

Programmiert eines der wichtigsten Computerspiele der 80er Jahre!

Der Spieler navigiert PacMan durch ein Labyrinth und schluckt dabei Kraftpillen. Wenn alle Kraftpillen geschluckt wurden, erreicht PacMan den nächsten Level. Geister streifen durch das Labyrinth und versuchen, PacMan zu fangen. Sollte PacMan von diesen berührt werden, wird er gefressen und das Spiel ist vorbei.

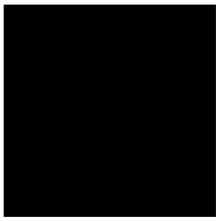
Erstellt diese Objekte auf dem Worksheet:



PacMan

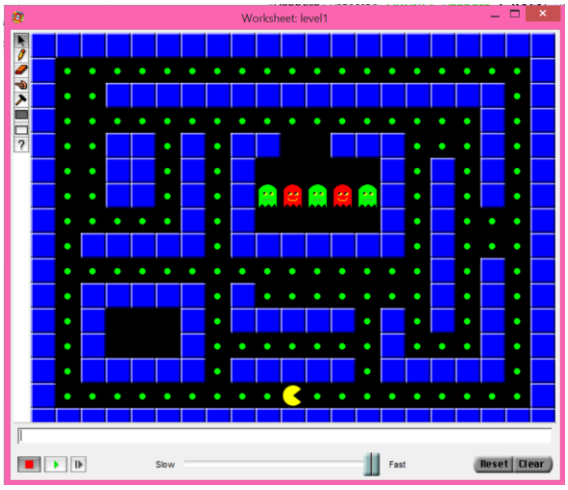
Zwei unterschiedliche Geister

Kraftpillen



Hintergrund

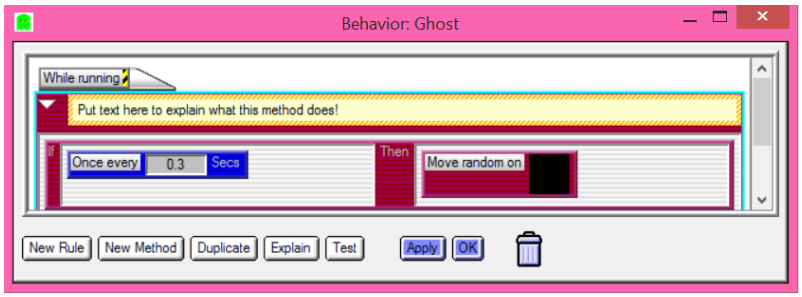
Mauer



Programmiert folgende Verhaltensweisen für eure Objekte

Schritt 1: Geister

Programmiert die Geister so, dass sie sich frei und zufällig bewegen können.



Schritt 2: PacMan

Programmiert ihn so, dass er mit den Pfeiltasten bewegt werden kann.

Schritt 3: PacMan darf nicht durch Wände laufen können

Arbeite mit deinem Nachbarn zusammen und findet gemeinsam heraus, wie ihr PacMan davon abhalten könnt, durch Wände zu laufen.

Schritt 4: Ermögliche es PacMan, Kraftpillen zu "essen"

Arbeite eventuell mit deinem Nachbarn zusammen und findet gemeinsam heraus, wie ihr PacMan Kraftpillen schlucken lassen könnt.

Schritt 5: Das Spiel endet, wenn PacMan direkt neben einem Geist ist

Vergesst nicht, dass die Simulation einen RESET benötigt, um das Spiel zu beenden.

Schritt 6: Teste das Spiel

Drückt den grünen Pfeil und spielt das Spiel.

- ✓ Bewegt sich PacMan in alle vier Richtungen?
- ✓ Bleibt PacMan auf dem Weg und kann die Wände nicht durchqueren?
- ✓ Bewegen sich die Geister frei und willkürlich?
- ✓ Schluckt PacMan die Kraftpillen?
- ✓ Wird das Spiel beendet, wenn sich PacMan neben einem Geist befindet?

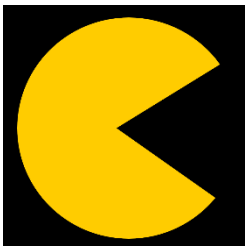
Anleitung für Schülerinnen/Schüler

Teil (Modul) 1/ Handout 1A

Programmiert eines der wichtigsten Computerspiele der 80er Jahre!

Der Spieler navigiert PacMan durch ein Labyrinth und schluckt dabei Kraftpillen. Wenn alle Kraftpillen geschluckt wurden, erreicht PacMan den nächsten Level. Geister streifen durch das Labyrinth und versuchen, PacMan zu fangen. Sollte PacMan von diesen berührt werden, wird er gefressen und das Spiel ist vorbei.

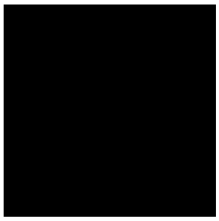
Erstellt diese Objekte auf dem Worksheet:



PacMan

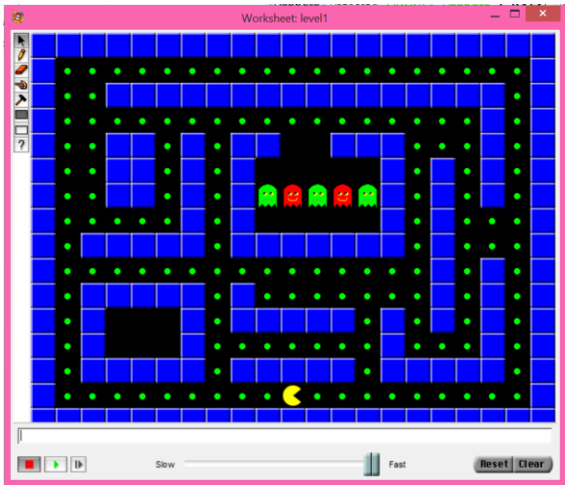
Zwei unterschiedliche Geister

Kraftpillen

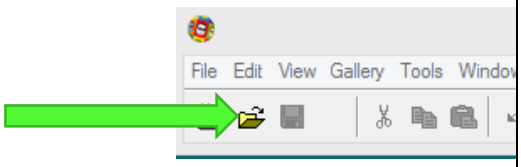
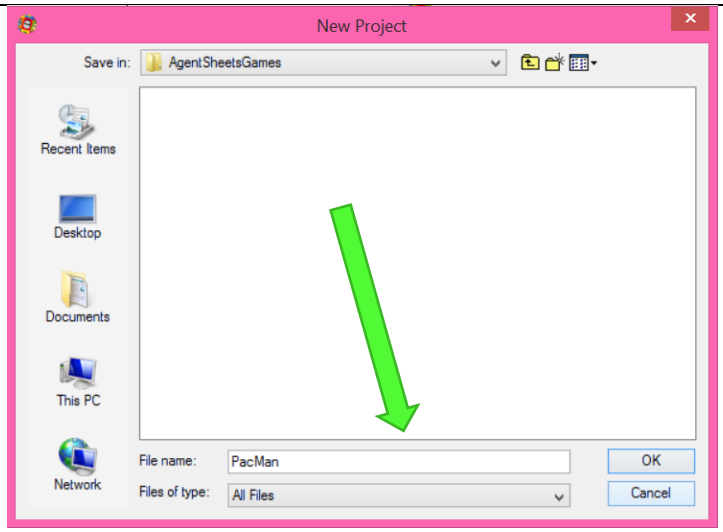
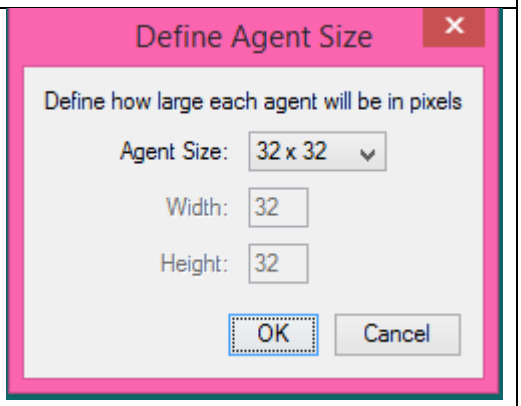


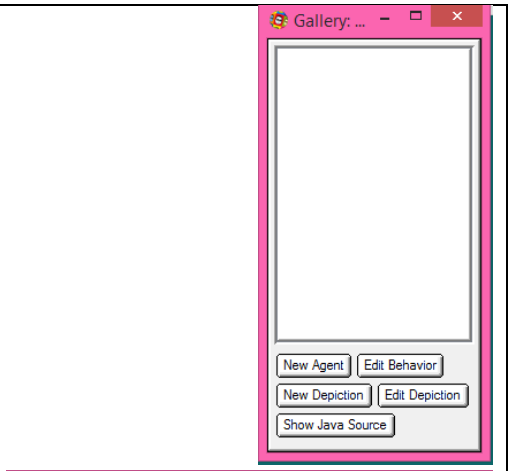
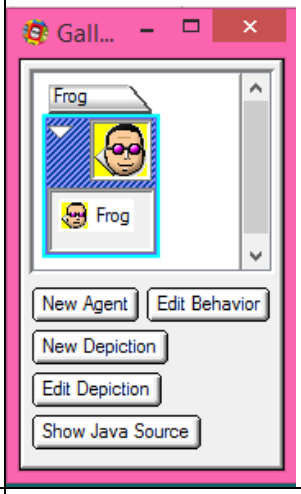
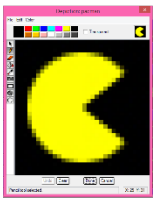
Hintergrund

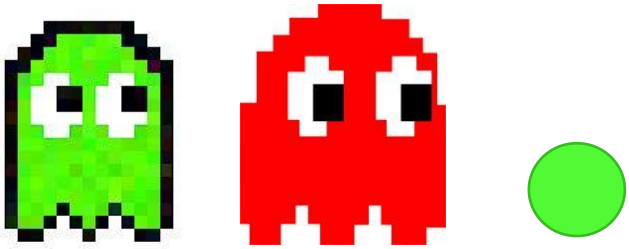
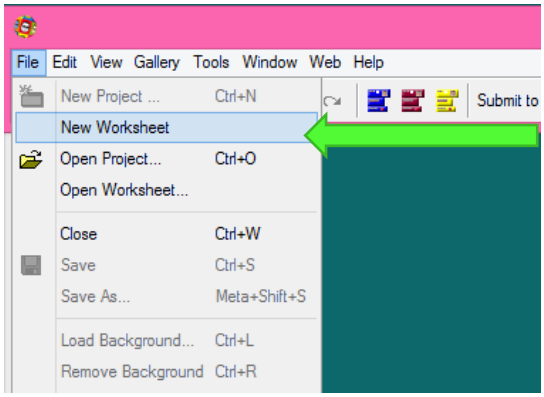
Mauer

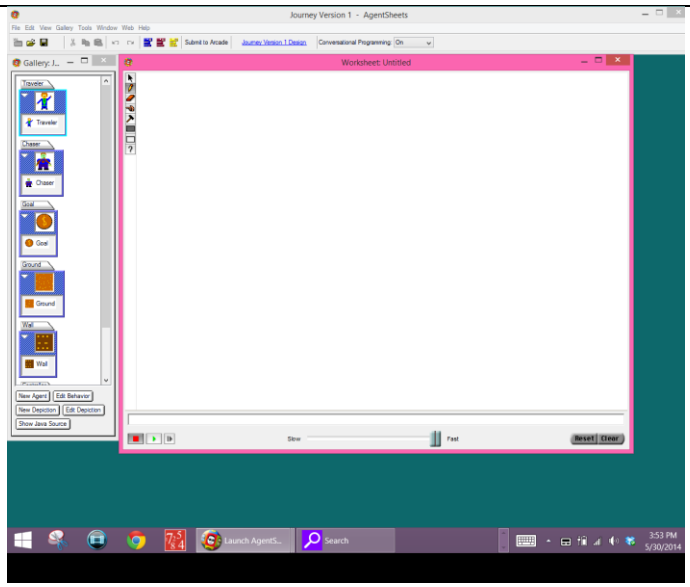
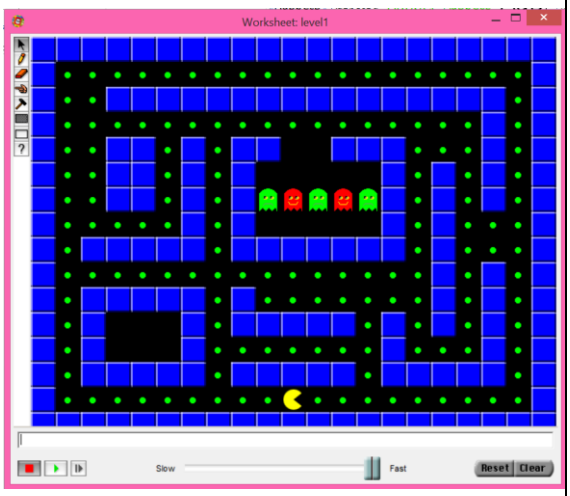


Zuerst eine ausführliche Anleitung, wie ihr AgentSheets startet, Objekte und das Spielfeld (Worksheet) bedient und dann PacMan, Geister und andere Objekte entwerft und programmiert.

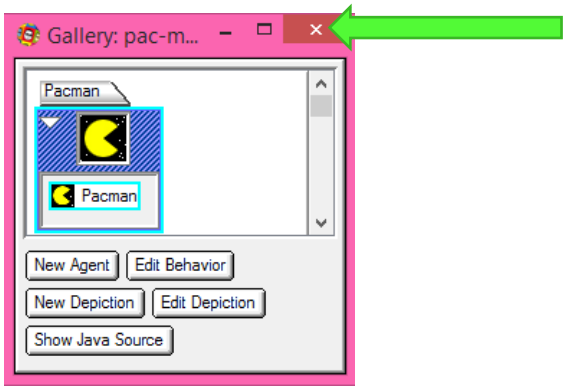
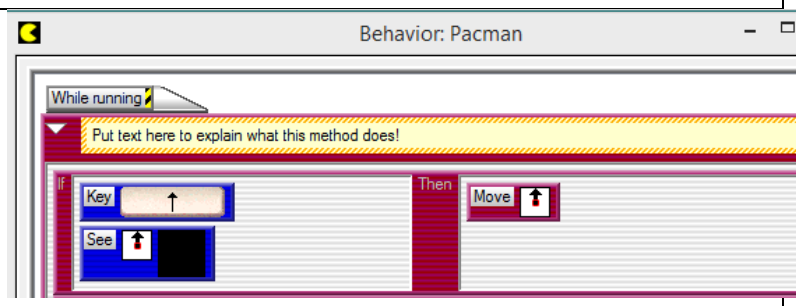
Schritt 1	Erstellt ein Spiel Klickt auf das Symbol ganz links	
Schritt 2	Benennt das Spiel Nennt es PacMan_xy und klickt OK (xy steht für euren Namen)	
Schritt 3	Objektgrösse Einstellung nicht ändern! Klickt OK	

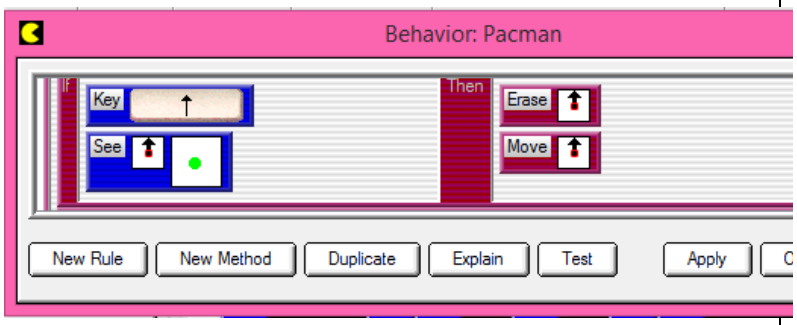
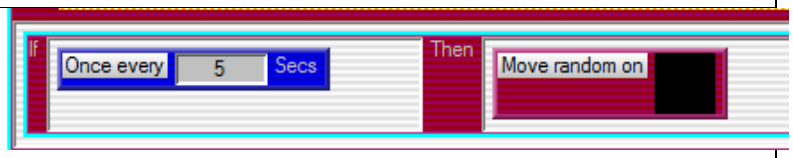
Schritt 4	Erstellt ein Objekt Klickt auf "New Agent" Nennt es "PacMan" Klickt OK	
Schritt 5	Zeichnet das Objekt Klickt "EDIT DEPICTION" Klickt "CLEAR", um das alte Bild zu löschen	*Klickt auf "Color"> "Mask Color">> wählt weiss, um den Hintergrund unsichtbar zu machen*  
Schritt 6	Zeichnet PacMan Klickt "DONE"	 So könnte euer Pacman aussehen. Aber ihr könnt ihn zeichnen, wie ihr wollt. Benutzt den Radierer (eraser), um etwas wegzuradieren oder drückt "clear" um

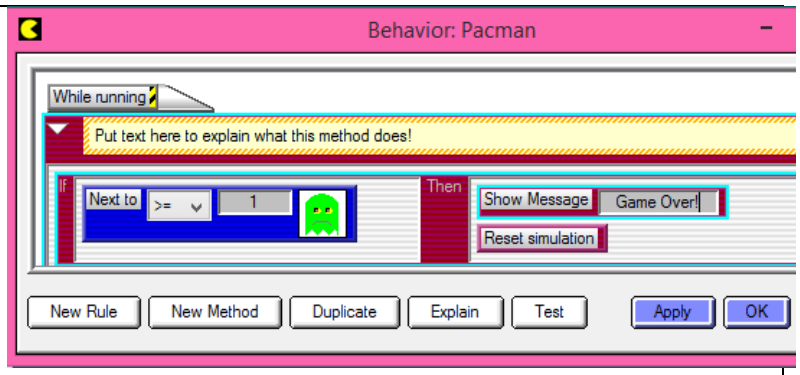
		das ganze Bild zu löschen.
Schritt 7	Zeichnet die restlichen Objekte	 Zwei Geister Kraftpille Zeichnet ein Geist-Objekt in grün, danach klickt ihr auf "*New Depiction*" und zeichnet einen weiteren Geist in rot in dasselbe Objekt. Ausserdem erstellst Du ein weiteres Objekt: die Kraftpille.   Weg Mauer
Schritt 8	Erstellt das Spielfeld (Worksheet) Klickt "File" -> "New Worksheet"	

Schritt 9	Zieht das Fenster grösser Es soll aber nicht den ganzen Bildschirm ausfüllen!	
Schritt 10	Benutzt folgenden Werkzeuge, um die Objekte zu platzieren. Stift (Pencil): Platziert ein Objekt nach dem anderen Graues Rechteck: Platziert Objekte auf grösserer Fläche (rechteckförmig).	**Wichtig** dabei ist, dass ihr niemals ein Hintergundobjekt (Weg, Wand) über ein anderes Objekt zeichnet. Wollt ihr zum Beispiel eine Wand ÜBER eine Kugel zeichnen, löscht zuerst die Kugel. 
Schritt 11	**SICHERT euer WOKRSHEET!!**	

Wir programmieren die Objekte

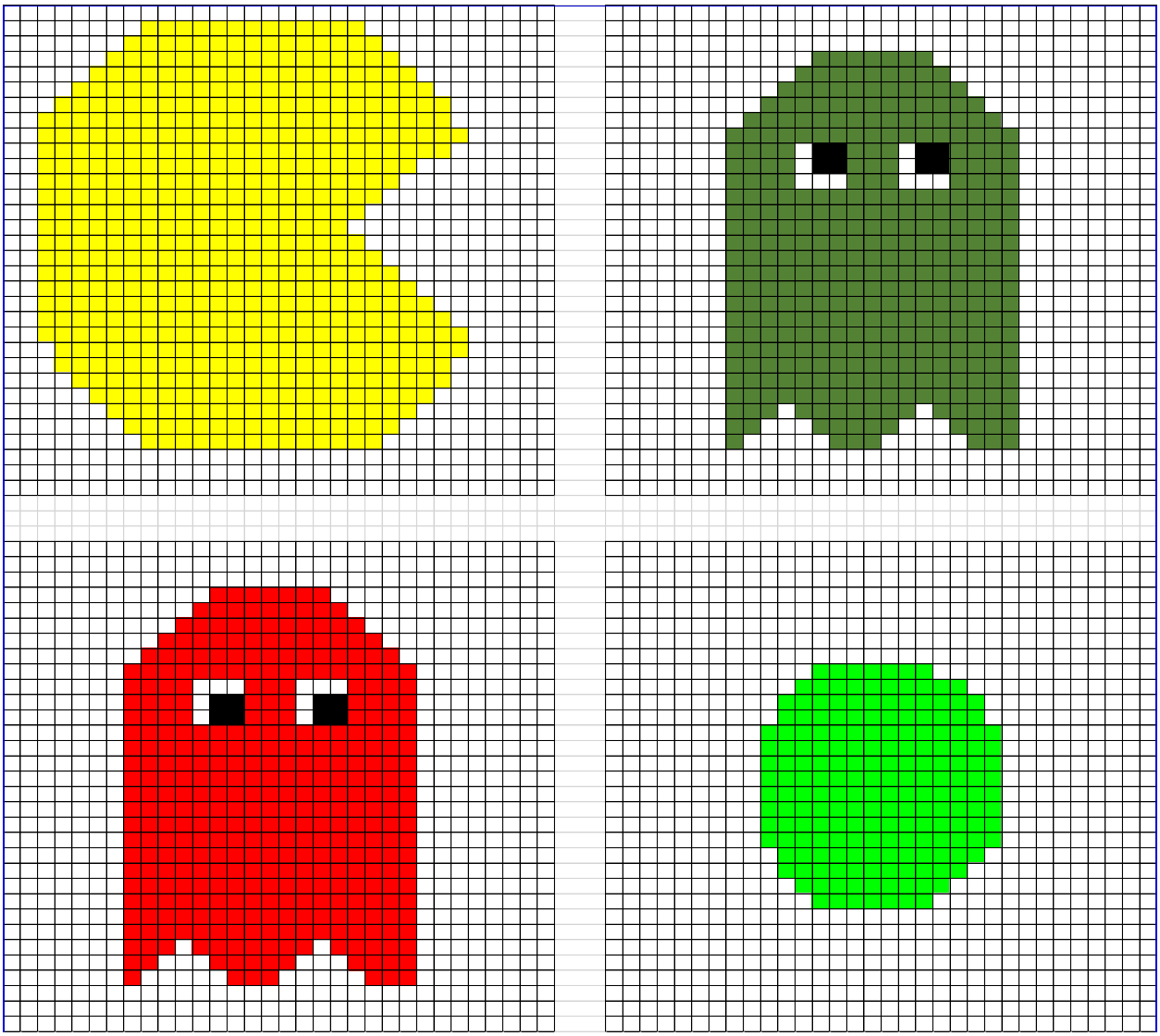
Schritt 12	Erstelle Verhaltensweisen für die Objekte Lies zuerst die Anleitung rechts Und danach Klickt "Edit Behavior"	Die Verhaltensweisen der Objekte werden "rules", Regeln, genannt. Regeln werden durch eine oder mehrere WENN (if) - DANN (then) Bedingungen dargestellt. Um PacMan mit den Pfeiltasten zu steuern, muss eine dieser Regeln lauten: "WENN die Pfeiltaste ↑ gedrückt wird, DANN soll sich PacMan nach oben bewegen". Wir brauchen auch eine Regel, damit sich PacMan nur auf dem Weg bewegt und nicht durch Wände laufen kann. Insgesamt werden hier vier Regeln programmiert, eine für jede Bewegungsrichtung von PacMan. 
Schritt 13	Verhaltensweisen erstellen Beim ersten Öffnen ist dieses Fenster leer. Ihr müsst die benötigten Bedingungen anklicken und links hineinziehen und die erwünschten Verhaltensweisen	 Schaut euch genau diese Regel an…sie besagt, WENN (if) die Pfeiltaste ↑ gedrückt wird UND ich einen Weg in dieser Richtung sehe, DANN (then) soll sich PacMan nach oben bewegen.

	rechts hineinziehen (drag & drop).	Erstellt nun alle Regeln für jede Richtung.
Schritt 14	Programmiere PacMan so, dass er die Kraftpillen schluckt	Denke zuerst kurz nach. Wenn PacMan direkt neben einer Kraftpille ist, soll sich die Kraftpille auflösen und so sieht es dann aus, als ob er sie geschluckt hätte. Schau dir genau die Regel an. PacMan sieht eine Kraftpille und diese verschwindet dann:  Die Regel sagt… WENN (if) die Pfeiltaste ↑ gedrückt wird und ich eine Kraftpille über mir sehe, DANN (then) lösche die Kraftpille über mir und bewege mich einen Schritt nach oben. Erstelle nun alle vier Regeln
Schritt 15	Programmiere den Geist Er soll ich willkürlich (nach dem Zufallsprinzip) bewegen.	 *Soll sich der Geist auch auf den Kraftpillen bewegen können? Oder auf dem Weg?* *Entscheide selbst und programmiere richtig! (im Bild oben bewegt sich der Geist nur auf dem Weg, nicht aber auf den Kraftpillen).*

Schritt 16	Erstelle eine Regel, um das Spiel zu beenden, wenn Pacman neben einem Geist ist Klicke auf PacMan und "*edit behavior*" Füge die Regeln hinzu	 Diese Regel sagt: WENN (if) ich neben einem oder mehreren Geistern bin DANN (then) zeige die Nachricht “Game Over!” und Reset die Simulation Vergiss nicht das RESET!!!

Schüler Handout 1B

Zeichenhilfe zur Objekterstellung



Anleitung für die Lehrperson

Teil (Modul) 2 – Wie der Geist den PacMan verfolgt

In diesem Teil des Projektes verändern die Schülerinnen und Schüler das Spiel und machen es spannender indem sich der Geist gezielt in Richtung des PacMan bewegt.

In dieser Lektion benutzen wir zwei Computational Thinking Muster:

Diffusion: PacMan hinterlässt einen imaginären Geruch. Der Duft verteilt sich gleichmässig entlang des Bodens und der Kraftpillen. Der Geruch *diffundiert* durch den Boden und die Kraftpillen. Er verteilt sich im ganzen Labyrinth, wobei die Intensität ja nach Entfernung zum PacMan zu- bzw. abnimmt.

Hill Climbing: Mit "hill climbing" lässt sich die Diffusion mathematisch beschreiben. Es werden 4 (oder 8) Richtungen auf die momentane Stärke des imaginären Geruchs verglichen. Die Richtung, in welcher der Geruch am stärksten ist, bestimmt die Bewegungsrichtung der Geister. Der Computer überprüft bei jedem Durchlauf erneut, wo der imaginäre Geruch am Stärksten ist und errechnet daraus, wie sich die Diffusion im Spielfeld ausbreitet. *Hill Climbing* ist ein weiteres "Computational Thinking" Muster (ein mathematischer Algorithmus), welches das physikalische Konzept der Diffusion als für den Computer verständlich umformuliert.

In diesem Modul lernen wir **OBJEKT-ATTRIBUTE** kennen.

Ein Objekt-Attribut ist eine VARIABLE, die LOKAL genutzt werden kann (von einem einzelnen Objekt). Bitte beachten Sie, dass AgentSheets darüber hinaus auch Simulationseigenschaften bzw. Variablen besitzt, die GLOBAL (von allen Objekten) genutzt werden können.

Anleitung:

Diskutieren Sie mit Ihren Schülerinnen und Schülern über das "Verhalten" der Geister.

Hier einige Anregungen:

- 'Verfolgt' der Geist PacMan tatsächlich? Warum oder warum nicht?
- Warum sollten wir das Spiel verändern, sodass der Geist tatsächlich PacMan verfolgt?
- Wie könnten wir das Spiel verändern, sodass der Geist PacMan verfolgt?

[Geben Sie den Schülerinnen und Schülern Zeit, um diese Aspekte mit ihrem Nachbarn zu besprechen. Sammeln Sie anschliessend die Ideen. Bitte denken Sie daran: Die Möglichkeit, dass Schülerinnen und Schüler neue Konzepte diskutieren, wird ihr konzeptionelles Verständnis dieser Konzepte fördern.]

Sie können das **Konzept der Diffusion** wie folgt in ihrer Klasse vorstellen: Stellt euch vor, PacMan verströmt einen Geruch, den der Geist riechen könnte … wäre es dann leichter für den Geist, PacMan zu finden? [Geben Sie den Schülerinnen und Schülern ein Beispiel, zu dem sie sich in Beziehung setzen können: Speck braten in der Küche; Geruch von frischem Kaffee; Haie, die einen Tropfen Blut im Meer ausfindig machen]

Erklären Sie: In Teil 1 dieses Projektes hat sich der Geist lediglich willkürlich auf dem Boden bewegt. Im Folgenden soll der Geist den PacMan mit **künstlicher Intelligenz** suchen. Hierzu wird ein *Computational Thinking* Muster eingesetzt, das "Hill Climbing" genannt wird.

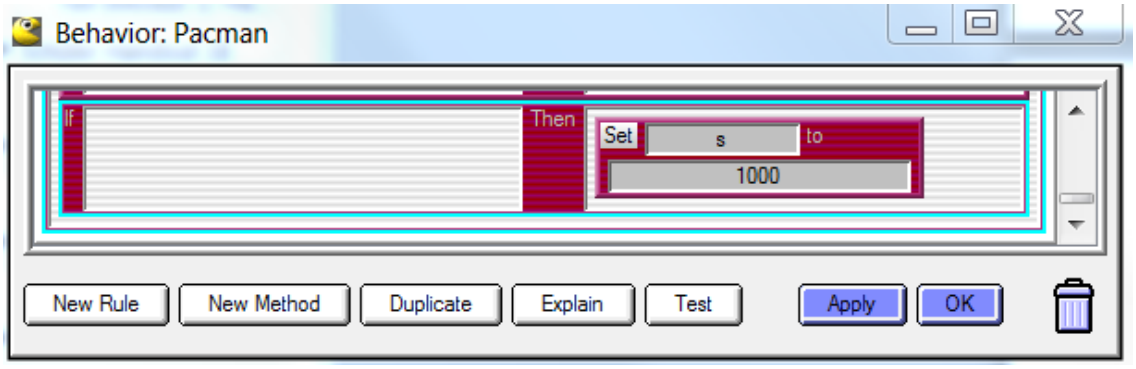
Stellen Sie sich vor, das PacMan-Objekt verströme einen Duft. Der Geruch wird von den Hintergrund-Objekten mittels „Diffusion" verbreitet. Diffusion ist ein physikalischer Prozess, bei dem Materie oder Dinge sich von der höchsten zur niedrigsten Konzentration bewegen und so einen Ausgleich / Durchmischung erreichen. Je näher man der Quelle des Geruchs kommt, desto grösser ist dessen Wert. „Hill climbing" ist ein Algorithmus, der zur Bestimmung der Richtung eingesetzt wird, aus welcher der Geruch am stärksten ist. Dies wenden wir bei den Geistern an, damit sie PacMan verfolgen.

Sie können Diffusion auch anhand eines Schachbrett-Musters erklären, auf welchem ein Initialwert eines zentralen Punktes (oder in unserem Fall des PacMan) mit der Entfernung systematisch abnimmt.

4	15	62		
15	62	250	250	
62	250	1000	250	
15	62	250		
	15			

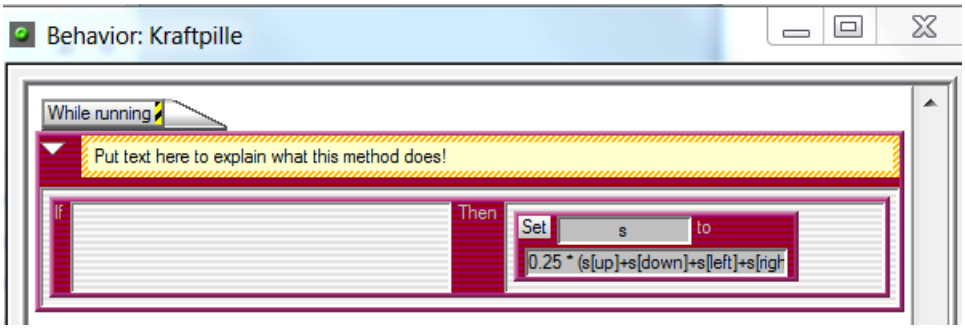
Die folgenden *Screenshots* von möglichen Codes sind für die Lehrperson gedacht, um diese mit der Klasse zu erarbeiten.

Egal, wohin sich PacMan bewegt, er ist immer die Quelle des Geruchs. Für diese Regel gibt es keine Bedingungen, deshalb sollte es immer die letzte Regel im PacMan sein.



Sowohl die Kraftpillen als auch der Weg verbreiten den Geruch, sodass die Geister die Richtung identifizieren können, aus welcher der stärkste Geruch kommt, wenn wir sowohl in die Kraftpillen aber auch in den Weg folgende Formel implementieren:

Set s to 0.25*(s[left]+s[right]+s[up]+s[down])



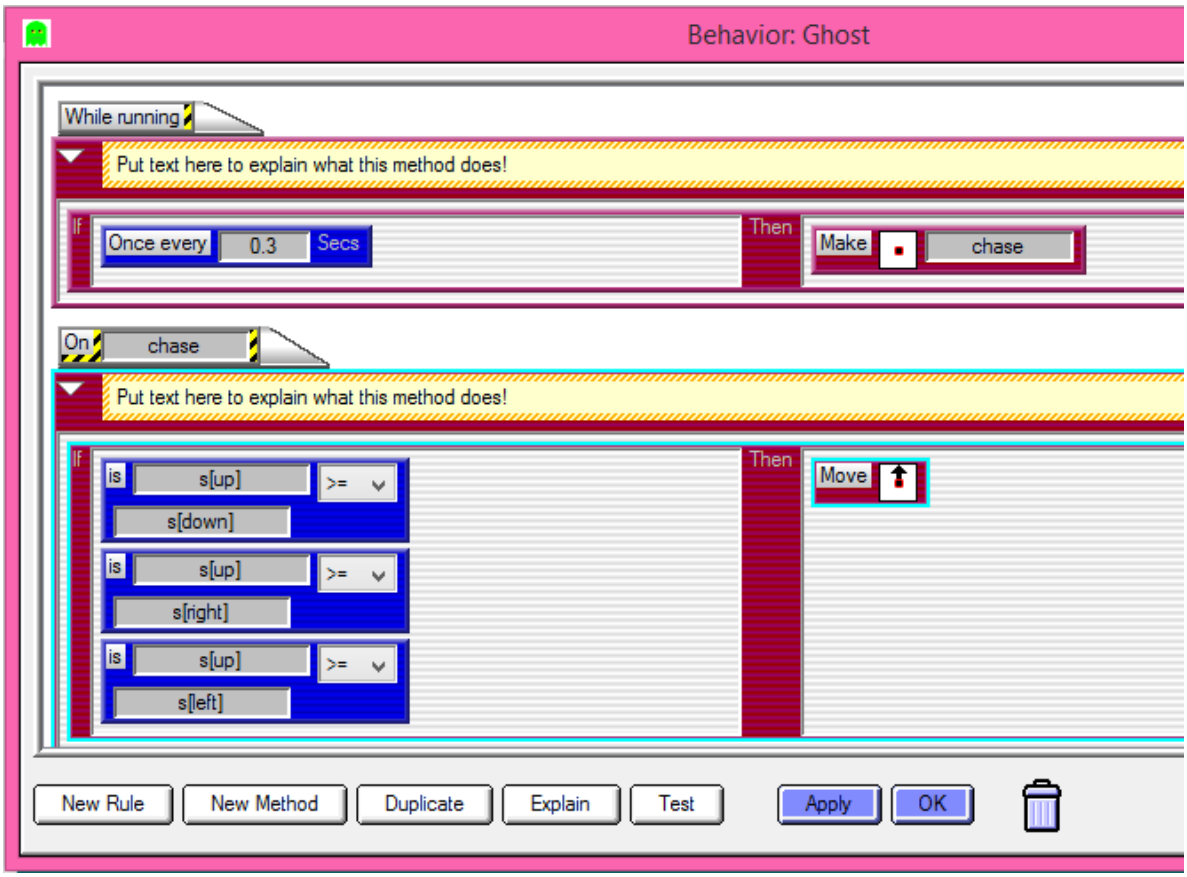
Anstelle s könnten wir die lokale Variable (Attribut) auch p oder mit einer anderen Abkürzung benennen.

METHODE

Um das Programm zum einen übersichtlicher zu machen und zum anderen Unterprogramme vielleicht an verschiedenen Stellen aufzurufen, kann man sogenannte "Methoden" definieren. Diese entsprechen auch dem Begriff der Funktionen. Für die Verfolgung von PacMan definieren wir eine Methode mit dem englischen Namen "chase" oder der deutschen Bezeichnung "verfolgen".

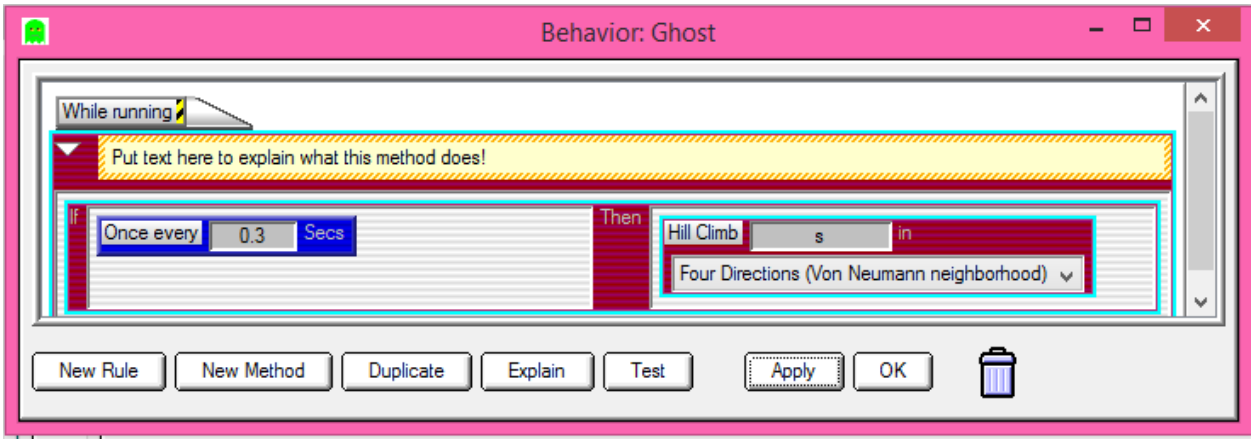
Das Kernstück jeder objektorientierten Überlegung bildet das Objekt. Allgemein kann man sagen, dass Objekte Attribute und Methoden enthalten. Dabei sind Attribute nur Variablen und Konstanten, die Werte aufnehmen können, und beschreiben damit das *statische Wesen* des Objektes. Im Gegensatz dazu gibt es die „Methoden“ die das gesamte *dynamische Verhalten* des Objektes widerspiegeln.

Alle 0.3 Sekunden wird der Geist dem Geruch folgen, indem er die Richtung bestimmt, aus welcher der stärkste Geruch kommt: entweder von oben, unten, rechts oder links. (Bitte beachten Sie, dass Sie lediglich die erste dieser vier Regeln in der Abbildung unten sehen können).



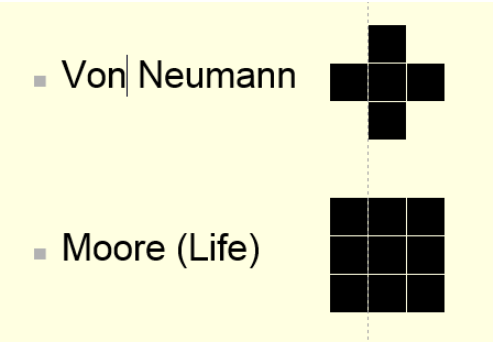
Ergänzung: Abkürzung für *Hill Climbing*

AgentSheets hat einen noch besseren Weg diesen Vorgang zu handhaben. Dieser sollte allerdings erst mit den Schülerinnen und Schülern geteilt werden, nachdem sie das Verfahren „Hill Climbing“ verstanden haben. Es lohnt sich, den „umständlicheren“ Programmiervorgang komplett durchzuführen, bevor den Lernenden die Abkürzung präsentiert wird. Der Einsatz dieses Codes erleichtert zukünftige Spielerweiterungen immens (z.B. wenn die Geister vor PacMan weglaufen sollen).



Diese Regel ist eine Abkürzung zur Überprüfung aller vier Dimensionen. Dies wird in der Regel als "*Von Neumann's Neighborhood*" definiert. Ausserdem könnten die Schülerinnen und Schüler die "*Moore Neighborhood*" auswählen und in acht Richtungen den Geruch überprüfen.

Anschliessend könnten Sie die Lernenden bitten, beide Programmiermethoden und beide Prozesse miteinander zu vergleichen und zu beschreiben.



MAP Funktionalität

Die Lernenden können mit Hilfe der *MAP-Funktionalität* die Diffusion farblich kennzeichnen und so sehen, tatsächlich passiert.

Lernenden Handout 2

Teil 2 – Die Gespenster jagen PacMan

Bis jetzt bewegt sich dein Geist nur willkürlich, entweder nur auf dem Boden oder auf dem Boden und den Kraftpillen … er jagt den PacMan nicht wirklich. Das wollen wir nun ändern!

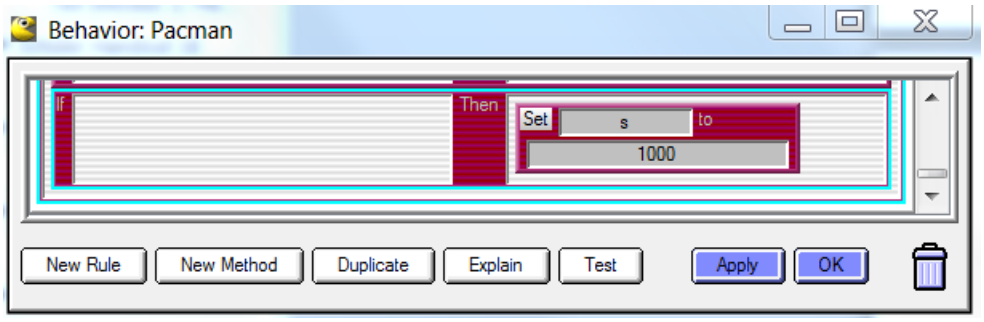
HILLCLIMBING Der Geist soll das PacMan-Objekt auf "intelligente" Art und Weise suchen (also nicht nach dem Zufallsprinzip). Stell dir vor, das PacMan-Objekt verströmt einen Geruch. „***Hill Climbing***“ ist ein Verfahren, auch Algorithmus genannt, um die Richtung zu finden, aus welcher der stärkste Geruch kommt. Es ist nur eine Annäherung an das optimale Ergebnis - Iteration ist hier der Fachbegriff. Wie der Name schon andeutet, kann man dieses Verfahren mit folgendem Bild vergleichen: Ein Bergsteiger will einen Gipfel erklimmen. Es ist neblig. Der Bergsteiger besitzt weder eine Karte, noch befindet er sich auf einem Weg. Die einzigen Hilfsmittel die er besitzt, sind ein Kompass und ein Höhenmesser. Um den Gipfel zu erreichen wird der Bergsteiger von seiner Position aus einen Schritt nach Norden machen, die Höhe feststellen und wieder einen Schritt zurück auf seine ursprüngliche Position gehen. Dieselbe Prozedur wiederholt er für die anderen drei Himmelsrichtungen. Der Bergsteiger hat nun für jede Himmelrichtung notiert, in welche Höhe er gelangen kann, wenn er einen Schritt in dieser Richtung geht und wird sich für die Richtung entscheiden, in die er die größte Höhe erreicht.

DIFFUSION Der Geruch wird von den Hintergrund-Objekten (Boden, Kraftpillen) mit Hilfe des Computational Thinking Musters „**Diffusion**“ verbreitet. Diffusion ist ein Konzept aus der Physik, Biologie, Ökolgie etc., in welchem sich Dinge, Moleküle oder Objekte von Bereichen mit der höchsten Konzentration zu Bereichen mit der niedrigsten Konzentration bewegen. Je näher man der "Quelle" des Geruchs ist, desto höher ist dort die Konzentration und desto grösser ist dort dessen Wert.

ATTRIBUTE Dieser Teil des Projekts führt in das Konzept von “Objekt-Attributen” ein. **Attribute** sind spezifische Objekt-Eigenschaften und -Informationen, die bei jedem Aufruf des Objektes abgefragt werden. Informatiker nennen diese Attribute eine **lokale Variable**.

Schritt 1:

Zuerst lass uns sicherstellen, dass PacMan einen Geruch absondert. Um das zu tun, brauchen wir ein Attribut „s“ für PacMan (Wir haben dem Attribut einen beliebigen Namen gegeben: „s“ für das englische Wort scent = Geruch). Wir fügen diese Regel am Ende aller PacMan-Regeln ein:



Diese Regel sagt zu PacMan: “Wenn ich sonst nichts tue, dann verströme ich einen Geruch in Höhe von s=1000 an der Stelle, wo ich mich gerade aufhalte.” Es ist wichtig, dass diese Regel NACH allen anderen Regeln für PacMan steht, also ganz am Ende der Liste, da sie in jedem Moment aufgerufen werden soll.

Schritt 2:

Nachdem der Geruch diffundiert oder sich ausbreitet, müssen wir den Durchschnittswert von s (Geruch von PacMan) in dem beschriebenen Teil des Hintergrunds ermitteln. Stell dir einfach vor, dass der Geruch aus allen vier Himmelsrichtungen (Norden, Süden, Osten und Westen) kommt. Der Geruch in der Mitte ist dann der Durchschnitt dieser vier Werte. Wie würdest du das programmieren?

Geruch verbreitet sich mittels der Kraftpillen

Das Objekt Kraftpille soll das in Schritt 2 beschriebene Verhalten haben, welches wir mit folgendem Algorithmus programmieren:

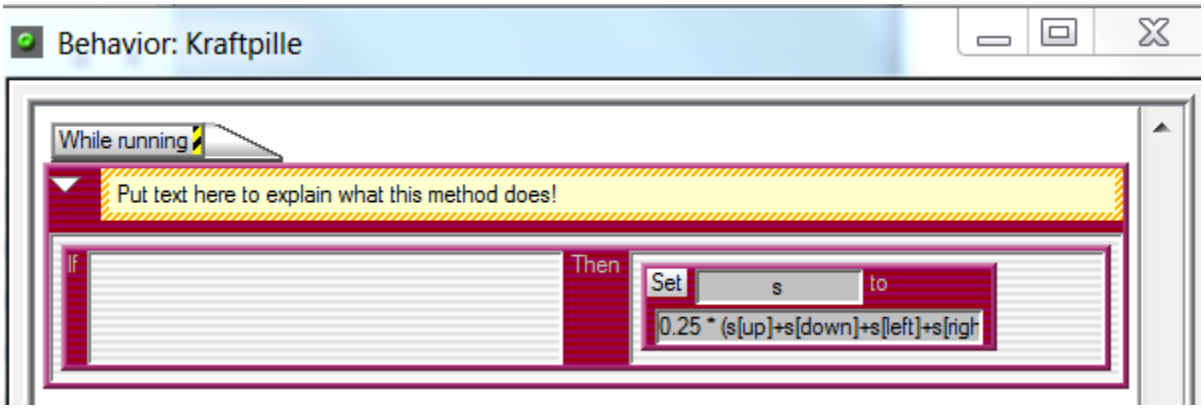
Set s to 0.25*(s[left]+s[right]+s[up]+s[down])

s steht für scent (Geruch) und die Formel ergibt einen Mittelwert aus dem Wert der umliegenden 4 Felder. Der Set-Befehl ersetzt den Initialwert von s (also 1000) hier mit dem Mittelwert. Um den Mittelwert zu ermitteln, addierst du die Werte aller 4

benachbarter Felder und dividierst sie durch die Anzahl 4. Dividieren durch 4 ist das gleiche wie multiplizieren mit 0.25.

4	15	62		
15	62	250	250	
62	250	1000	250	
15	62	250		
	15			

Achte in der Gleichung auf die Klammern. Es gibt runde “(” und eckige Klammern “[”.



Jetzt programmiere analog den Hintergrund!

Schritt 3:

Damit die Geister wissen, in welche Richtung sie laufen müssen, müssen sie berechnen, aus welcher Richtung der stärkste Geruch kommt. Es gibt dafür einen Begriff im Englischen, das sogenannte „HILL CLIMBING”. Nach diesem Schema checkt der Geist alle Richtungen (oben, unten, rechts und links). Wo auch immer der stärkste Geruch herrscht, in diese Richtung soll sich PacMan bewegen.

Wir werden eine METHODE für den Geist erstellen, damit er einer ganzen Reihe von Regeln folgt.

METHODE

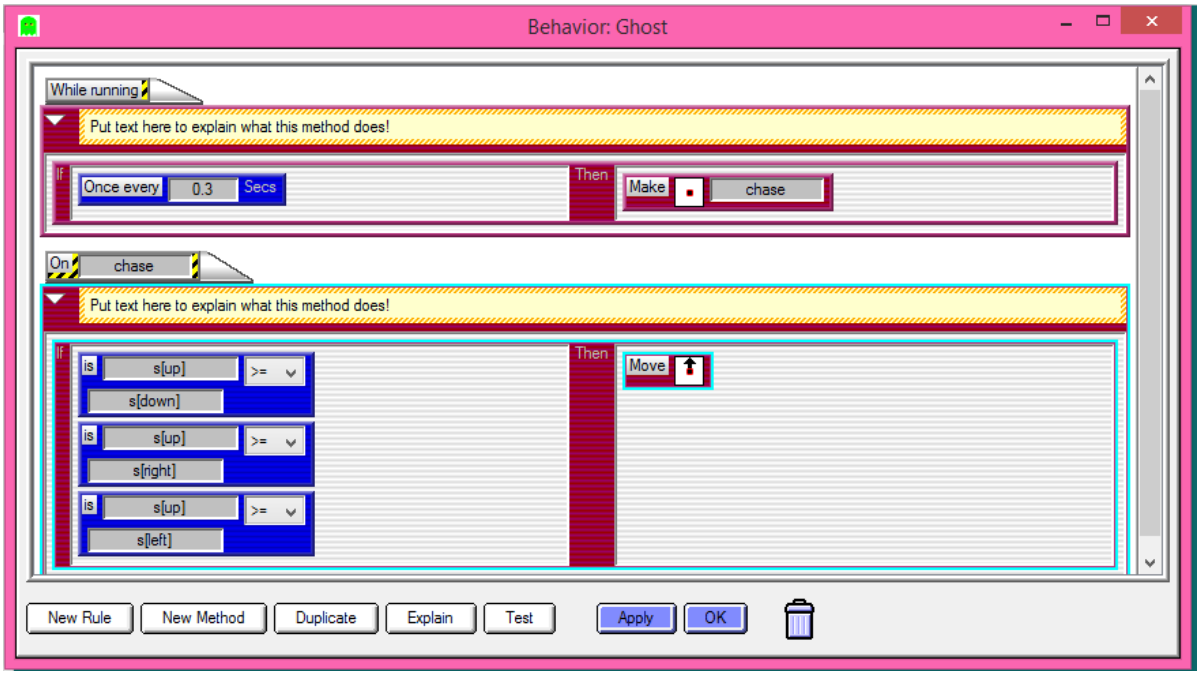
Um das Programm zum einen übersichtlicher zu machen und zum anderen Unterprogramme vielleicht an verschiedenen Stellen aufzurufen, kann man sogenannte "Methoden" verwenden. Für die Verfolgung von PacMan definieren wir eine Methode mit dem englischen Namen "chase" oder der deutschen Bezeichnung "verfolgen".

Schau dir die Programmierung unten an. Sie sagt folgendes:

ALLE (im Englischen "once every") 0.3 Sekunden, führe die "chase" Methode aus (make me "chase" - chase steht für verfolgen).

Die "chase" Methode beinhaltet:

WENN (im Englischen if) der Geruch über dir grösser oder gleich aller anderen Gerüche in deiner Umgebung (oben, unten, links oder rechts) ist, DANN (im Englischen then) bewege dich nach oben.



Füge jetzt die restlichen Regeln ein, sodass der Geist weiss, was er machen muss, wenn der Geruch unter ihm (s[down] grösser ist … Was passiert, wenn der Geruch auf der linken Seite grösser ist? Was ist mit dem Geruch auf der rechten Seite?

Starte dein Spiel und überprüfe, ob der Geist PacMan verfolgt! Wenn es nicht funktioniert, ist nun die beste Zeit für die Problemsuche.

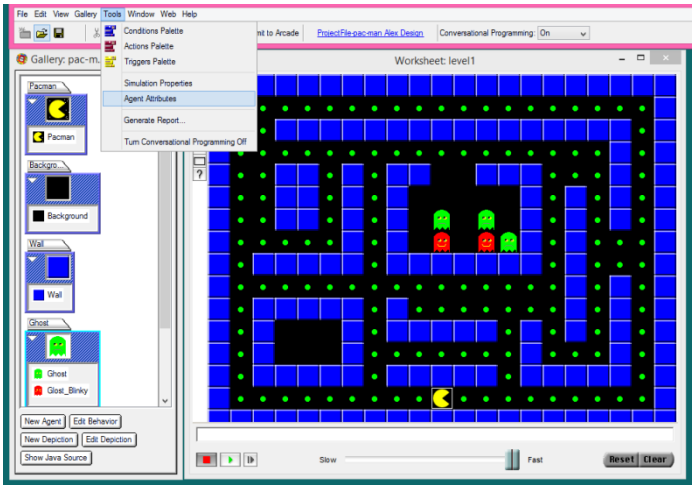
Überprüfe folgendes:

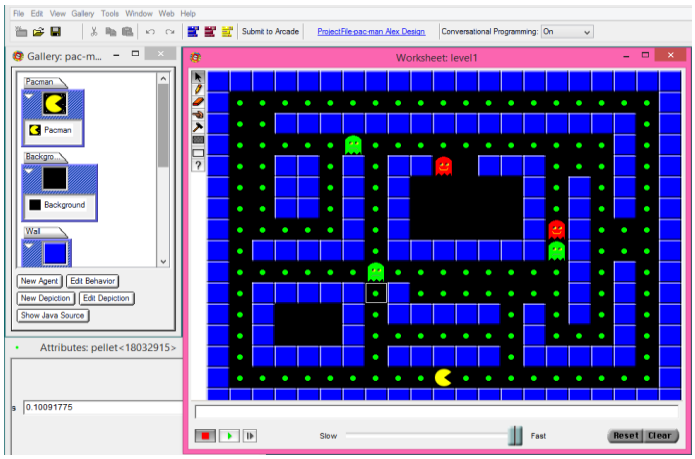
- Die Reihenfolge und den Standort der Regeln
- Den Einsatz der Methode
- Die Verwendung der runden und eckigen Klammern

Lernenden Handout

Diffusion und Hill Climbing / Testen

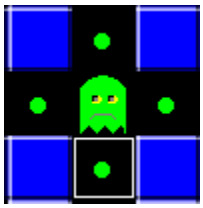
Um zu ermitteln, was in deinem Spiel passiert, ist es manchmal hilfreich, sich die Objekt-Attribute anzuschauen. Gehe auf dein Worksheet und klicke „run“ bis sich die Geister aus der Box bewegt haben und klicke dann auf Stopp. Setze das Spiel an dieser Stelle nicht zurück. Dein PacMan hat nun seinen Geruch verströmt. Du kannst seinen Geruch sehen (den Wert von s), wenn du auf irgendeine Stelle des Hintergrundes klickst und dann in der Menüübersicht auf *Tools>>Agent Attributes* klickst.





Es wird ein Kasten sichtbar werden, der den Wert des Attributs deines Objekts zeigt.

In diesem Kasten siehst du den Wert des Geruchs, links vom Geist. Wenn du die Attribute der vier Kästchen überprüfst, die um den Geist herum sind (oben, unten, links und rechts), und dann das Spiel erneut startest, kannst du überprüfen, ob der Geist das tut, was er soll.



Wenn er das nicht tut, überprüfe die Regeln und Methoden. Dinge, die du hierbei berücksichtigen solltest:

- **Rechtschreibung**
- **Runde und eckige Klammern**
- **Reihenfolge der Regeln**

Teil 3 – Das Spiel ansprechender gestalten

Anleitung für die Lehrpersonen

Zählen und Daten übermitteln (broadcast)

Folgende Fragestellung ist gegeben:

> *Unser Spiel endet bisher dann, wenn PacMan von einem Geist erwischt wird. Jetzt müssen wir eine Möglichkeit finden, in der das Spiel endet, wenn alle Kraftpillen geschluckt worden sind.*

Lassen Sie die Schüler einige Minuten die neuen Ideen diskutieren.

- Wer sammelt ein (prüft, ob noch Kraftpillen vorhanden sind)
- Was beendet das Spiel?
- Welche Regeln müssen dafür verändert werden?

Die Schüler könnten Probleme haben, genau darzulegen, wer das Einsammeln überwachen sollte. Stellen sie in diesem Fall die Möglichkeit vor, ein Kontrollobjekt hinzuzufügen, welches für das Zählen der verbliebenen Kraftpillen verantwortlich ist. Weisen Sie die Schüler an, genau über die Programmierschritte nachzudenken, damit sie diese später schneller erarbeiten können.

Schüler Handout 3

Das Spiel ansprechender gestalten

Zählen und Daten übermitteln (*broadcast*)

Um das Spiel anspruchsvoller zu machen, soll PacMan alle Kraftpillen im Labyrinth schlucken, um das Spiel zu gewinnen. Sind nicht alle Kraftpillen geschluckt, ist der Level nicht geschafft.

Dazu werden wir neue SIMULATIONSEIGENSCHAFTEN (simulation properties) nutzen. Mit diesen lassen sich Informationen zwischen den Objekten austauschen.

Du kannst dir aussuchen, ob PacMan als “Controller” funktionieren soll oder ein neues Objekt dafür verantwortlich ist, um die Kraftpillen zu zählen bis alle geschluckt worden sind.

Schritt 1: Die Kraftpillen zählen und ermitteln, ob du gewonnen hast.

Stelle dir folgende Unterhaltung vor…

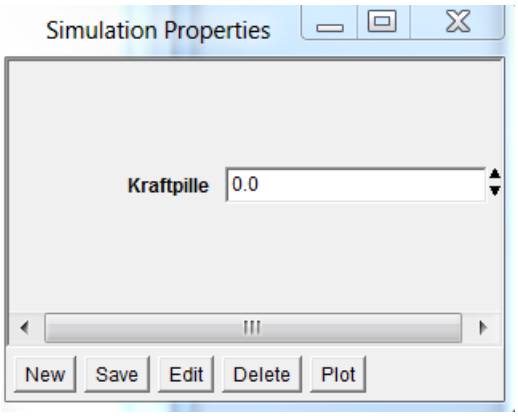
Der Lehrer hat der Klasse einen Arbeitsauftrag erteilt und will wissen, ob alle fertig sind. Sie/Er sagt zu der Klasse: "Hebt eure Hand, wenn ihr noch arbeitet." Einige Hände gehen hoch - fünf Schüler arbeiten noch. "Okay, dann arbeitet jetzt weiter".

Ein paar Minuten später, macht sie/er es wieder. Sie/Er sagt zu der Klasse: "Hebt eure Hand, wenn ihr noch arbeitet." Einige Hände gehen hoch - zwei Schüler arbeiten noch. "Okay, dann arbeitet jetzt weiter".
Ein paar Minuten später wird dieselbe Frage gestellt: "Hebt eure Hand, wenn ihr noch arbeitet." Dieses Mal gehen keine Hände nach oben. "Wunderbar, da jeder fertig ist, legt eure Bücher nun weg"

So ähnlich wird auch unser neuer Programmierauftrag funktionieren. Der Controller soll so lange zählen, bis er bei null angekommen ist (wie im Klassenzimmer, wenn keine Hände mehr hochgehen und der Lehrer weiss, ihr seid fertig). Wenn die Kraftpillen "hören", dass der Controller fragt (Befehl: *Daten übermitteln = broadcast)*, reagieren die Kraftpillen darauf (heben in unserer Vorstellung die Hand). Der Controller zählt die Kraftpillen. Wenn die Antwort grösser Null ist, geschieht nichts, das Spiel geht weiter. Wenn die Antwort Null ist (was bedeutet, dass es keine verbleibenden Kraftpillen mehr im Labyrinth gibt), endet das Spiel.

Zuerst erstellen wir eine Simulationseigenschaft (simulation property) und nennen sie zum Beispiel "Kraftpillen". Damit werden die Kraftpillen gezählt. Gehe dazu zu "Tools" >>"Simulation Properties">>"New". Schreibe "Kraftpillen" und klicke *speichern*.

Um auf genau diese Eigenschaft zu verweisen, benutzen wir das @ Symbol. Da wir hier auf die Kraftpillen verweisen wollen, müssen wir @Kraftpillen schreiben.



Simulation Properties:

Wie funktionieren "Simulation Properties" (Simulationseigenschaften)?

In der “While Running” Methode, setzt der Controller die Simulationseigenschaft "Kraftpillen" auf Null. Dann sendet er ein Signal an alle Kraftpillen. Dieses Nachfragen wird als Zählen bezeichnet. Danach prüft der Controller mit der "pruefe_gewonnen" Methode, ob das Spiel gewonnen wurde. Dies ist dann erfüllt, wenn es keine Kraftpillen mehr im Labyrinth gibt, was durch die Eigenschaft darüber ausgedrückt wird.

Das **Controllerverhalten** muss drei Abfragen machen.

1: Die Anzahl der Kraftpillen soll auf null gesetzt sein (wie wenn der Lehrer sagt, "Hände runter”)

Setze @Kraftpillen auf Null

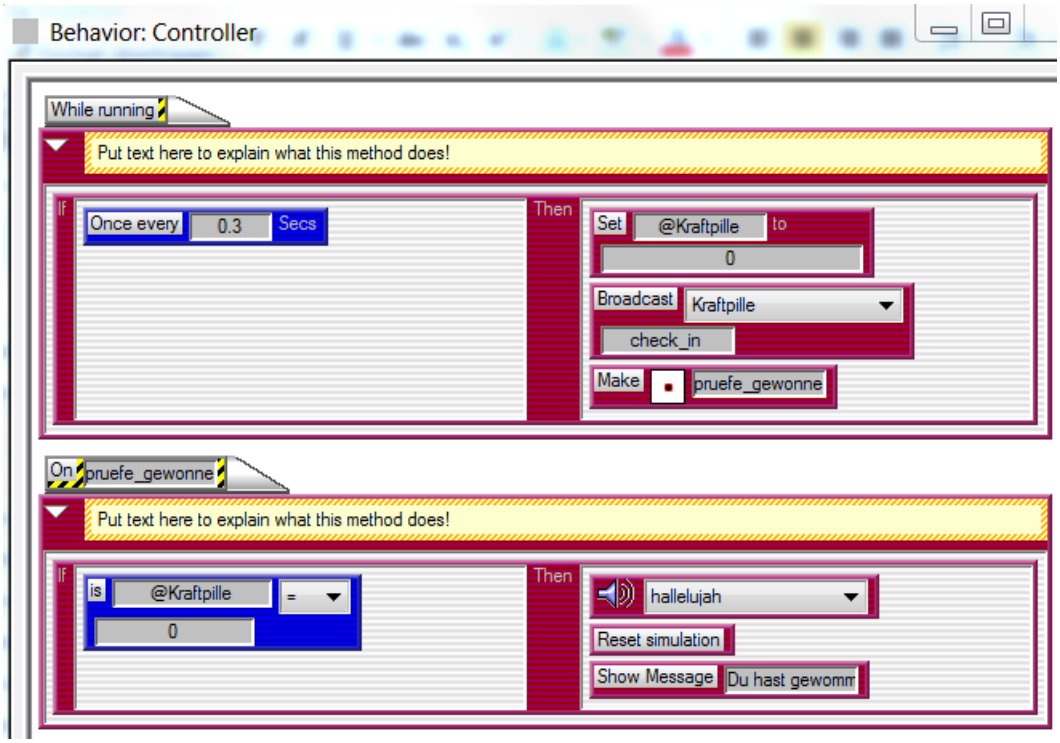
2: Frage die Kraftpillen, ob noch welche im Labyrinth sind.

Daten übermitteln (broadcast). Der Controller hat die Methode: pruefe_gewonnen

3: Benutze das Abzählen der Kraftpillen, um zu sehen, ob du gewonnen hast.

Make myself pruefe_gewonnen

So sollten die Regeln für deinen PacMan bzw. den Controller aussehen:



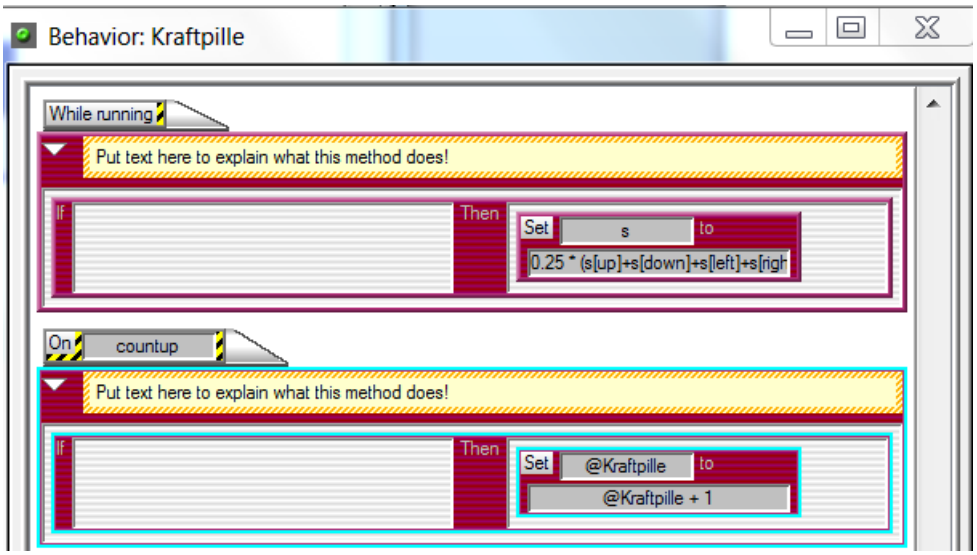
Änderung im Verhalten der Kraftpillen: Der Controller hat die Kraftpillen aufgefordert, die “pruefe_gewonnen” Methode auszuführen, aber sie wissen noch gar nicht, was damit gemeint ist. Das müssen wir nun programmieren.

Die Regel soll ermöglichen, dass die Kraftpillen auf die Frage "pruefe_gewonnen" (Daten übermitteln mit dem broadcast Befehl) vom Controller reagieren, um die Simulationseigenschaft @Kraftpillen zu aktualisieren.

Diese Änderung wird als separates Verfahren programmiert; es ist nicht Teil der "while running"- Methode, da sie nur dann benutzt wird, wenn vom Controller danach gefragt wird.

Wird die "pruefe_gewonnen"- Methode aufgerufen, soll jede verbleibende Kraftpille den Wert der @Kraftpillen Simulationseigenschaft erhöhen. Sind keine Kraftpillen mehr im Labyrinth, wird der Wert Null sein, was der Controller dann erkennt und das Spiel als gewonnen erklärt.

Im Folgenden ist das Programm der Kraftpillen gezeigt:


Behavior: Kraftpille
While running
Put text here to explain what this method does!
If
Then
Set
s
to
0.25 * (s[up]+s[down]+s[left]+s[righ
On
countup
Put text here to explain what this method does!
If
Then
Set
@Kraftpille
to
@Kraftpille + 1

Lernenden Handout: Probleme beheben beim PacMan

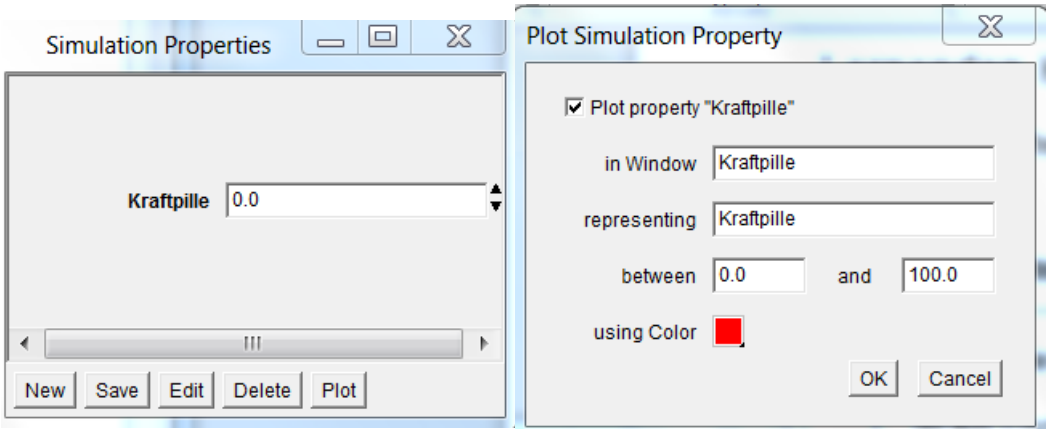
Zählen und Daten übermitteln (broadcast)

Häufigste Probleme:

1. Ist ein Controller Objekt auf dem Worksheet?
2. Hast du @Kraftpillen eingetragen?
3. Hast du immer auf das richtige Objekt verwiesen?

Ausführliche Problembehebung:

Um herauszufinden, warum etwas in dem Spiel passiert, ist es sinnvoll, die "simulation properties" anzuschauen. Öffne dazu die "simulation property box" (Tools>>Simulation Property). Klicke auf "property" und danach auf "Plot". Das müsste dann so aussehen:



Klicke auf "Plot Property ‘Kraftpille’". Ändert die Einstellung so, dass 0 links steht und rechts die Zahl genau der Anzahl der Kraftpillen in dem Labyrinth entspricht.

Damit wird ein Graph erstellt, der dir anzeigt, was im Hintergrund des Spiels passiert, wenn du es spielst. Mit dieser Information lässt sich leichter herausfinden, wo ein Problem liegt. Geht beispielsweise die Anzahl der Kraftpillen nicht bis null, muss ein Problem bei der ‘pruefe_gewonnen’ Methode liegen oder dem Daten übermitteln (broadcast). Geht die Anzahl der Kraftpillen auf null, aber das Spiel wird nicht beendet, gibt es ein Problem beim "game ending" Kommando des Controllers.

Lernenden Handout 4a: PacMan schaut in die richtige Richtung

Bevor du damit anfängst:

PacMan sollte alle Kraftpillen schlucken und die Geister sollten sich zufällig oder mittels *hill-climbing* (also zielgerichtet) bewegen. Das Spiel ist verloren, wenn PacMan von einem Geist gefressen wird. PacMan und die Geister können nicht durch Wände laufen.

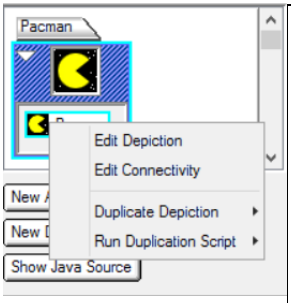
Beschreibung der neuen Aufgabe:

PacMan soll in die Richtung schauen, in die er läuft.

Was müssen wir machen:

- Brauchen wir ein neues PacMan Objekt?
- Brauchen wir neue Regeln für PacMan?

Wenn du jetzt glaubst, wir brauchen ein neues Pacman-Objekt, denke noch einmal kurz nach. Da jeder PacMan die Regeln ausführen kann, brauchen wir kein komplett neues Objekt, wir müssen nur sein Aussehen (*depiction*) anpassen.



Klicke die rechte Maustaste auf das kleine Bild von PacMan. Klicke bei "options" auf “duplicate depiction” und “run duplication script” um herauszufinden, wie du die drei anderen PacMan Gesichter darstellst. Gib ihnen allen unterschiedliche Namen, damit du nicht durcheinander kommst. Wenn du alles richtig gemacht hast, sollte es wie rechts im Bild aussehen.

Wenn alle vier PacMan Figuren erstellt sind, müssen wir die Regeln so anpassen, dass PacMan je nach Richtung sein Gesicht ändert. Teste nun dein Programm und prüfe, ob PacMan immer in die richtige Richtung schaut.

Lernenden Handout 4b: PacMan bewegt sich, ohne anzuhalten

Bevor du damit anfängst:

PacMan sollte alle Kraftpillen schlucken und die Geister sollten sich zufällig oder mittels *hill-climbing* (also zielgerichtet) bewegen. Das Spiel ist verloren, wenn PacMan von einem Geist gefressen wird. PacMan und die Geister können nicht durch Wände laufen. Pacman schaut immer in seine Bewegungsrichtung.

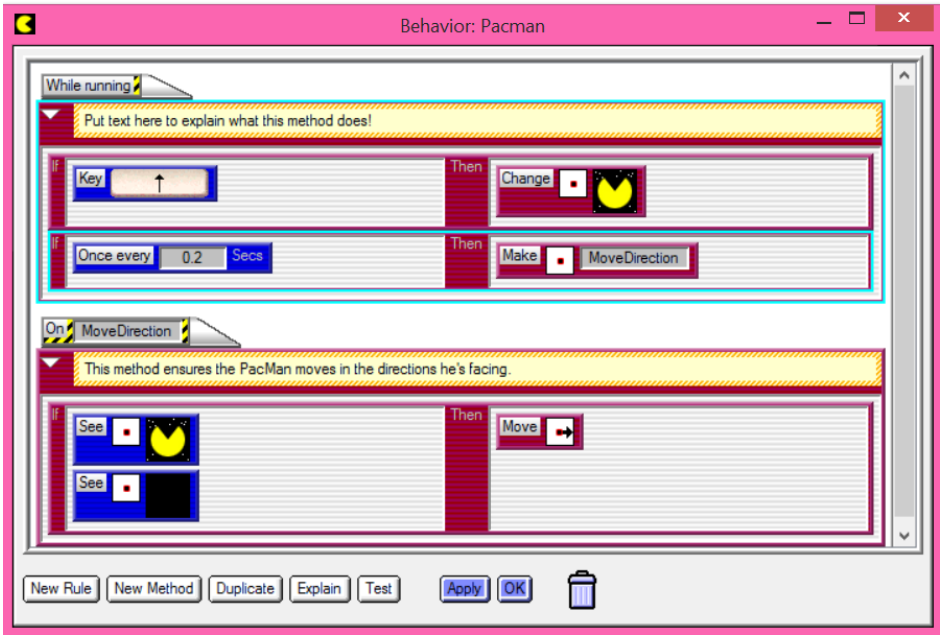
Beschreibung der neuen Aufgabe:

PacMan bewegt sich im Labyrinth ohne anzuhalten.

Diese Aufgabe hilft dir, anzufangen, verrät dir aber nicht alle Regeln. Wiederhole die Regel, die du unten siehst: Sie sagt, wenn der Hochpfeil gedrückt wird, wechsle das Aussehen zur Hochgehen Darstellung. Alle 0,2 Sekunden macht PacMan eine Bewegung.

Wenn die "MoveDirection " Methode aufgerufen wird, macht PacMan folgendes:

WENN (if) ich mich selbst nach oben blicken sehe UND ich sehe einen Weg, DANN (then) bewege ich mich nach oben.



Aber es muss noch mehr programmiert werden:

Schritt 1: Programmiere eine Regel, die PacMan sagt, was er zu tun hat, wenn er hochgehen soll und eine Kugel sieht.

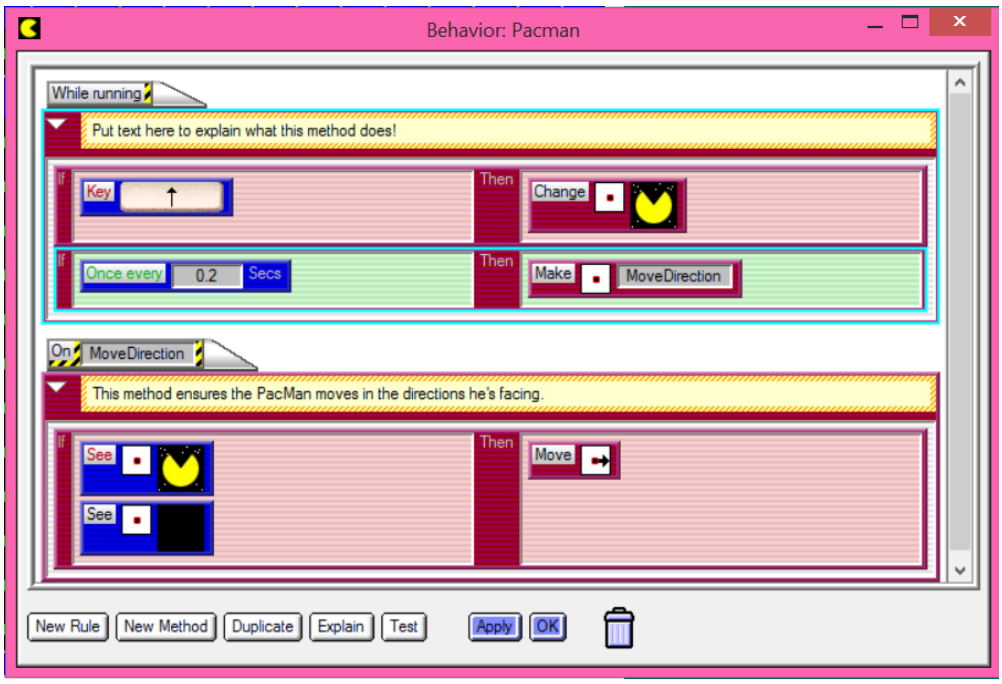
Schritt 2: Programmiere dasselbe für alle Richtungen.

Schritt 3: Teste dein Spiel. (Hinweis: Ist sichergestellt, dass PacMan seinen Geruch überall hinterlässt?)

Klicke auf PacMan und starte das Programm. Benutze verschieden Farben, um zu sehen, welche Regel richtig läuft und welche nicht. Im unteren Fall ist die erste Regel rot, das bedeutet, dass der Hochpfeil nicht gedrückt wurde.

Die nächste Regel ist grün, was bedeutet, dass Pacman sich alle 0,2 Sekunden einen Schritt weiter bewegt.

Die "MoveDirection" Methode ist rot. Das passiert, wenn eine oder alle Regeln darüber nicht erfüllt wurden. PacMan "sieht" nicht, dass er hochlaufen soll und /oder sieht keinen Weg. Daher ist diese Regel falsch und wird nicht ausgeführt.



Lernenden Handout 4c: Die Zauberpille

Bevor du damit anfängst:

PacMan sollte alle Kraftpillen schlucken und die Geister sollten sich zufällig oder mittels *hill-climbing* (also zielgerichtet) bewegen. Das Spiel ist verloren, wenn PacMan von einem Geist gefressen wird. PacMan und die Geister können nicht durch Wände laufen. Pacman schaut immer in seine Bewegungsrichtung und soll sich eigenständig weiterbewegen.

Beschreibung der neuen Aufgabe:

Zauberpillen sollen auf dem *worksheet* platziert sein.

Die Zauberpillen geben Pacman die Fähigkeit, die Geister zu schlucken. Die Geister werden dann blau und rennen vor PacMan davon.

Diese Aufgabe hilft dir anzufangen, verrät dir aber nicht alle Regeln.

- Du brauchst ein neues Objekt (Zauberpille)
- Brauchst du auch ein neues Objekt für die blauen Geister?
- Wenn die Geister PacMan jagen, hat er einen Geruchswert 1000. Was würde passieren, wenn PacMan einen Geruchswert -1000 hätte? Wie kannst du diesen neuen Wert eintragen? Wie kannst du diesen Wert nur eine bestimmte Zeit gültig machen?
- Hinweis: Benutze am besten die **hill climbing** Aktion für das Riechen.