Dokumentation zum Wumpus

Grundlagen zum Modellieren

Um das Weltbild modellieren zu können, wurde zunächst eine Klasse „Field“ erstellt. Diese Klasse repräsentiert ein Feld von dem Spiel. Ein „Field“ setzt sich zusammen aus dem Punkt, wo es sich auf dem Spielfeld befindet, dem spekulierten Typen, welcher zunächst angenommen wird, und einer Menge von Typen, die nachträglich hinzugefügt werden können. Der Typ eines Feldes kann „Gold, Empty, Visited, Breeze, Stench, Player, Start, Wumpus, Hole oder Wall“ sein.
Der Feldtyp „Empty“ zeigt an, dass der Spieler zunächst keine weiteren Informationen über dieses Feld besitzt.
Mit „Visited“ wird ein Feld versehen, auf welchem sich der Spieler bereits befand.
Der Typ „Breeze“ wird verwendet, um gespürte Luftzüge darstellen zu können.
Neben dem Luftzug gibt es dann noch das „Hole“, welches eine Falle darstellt, und den Typen „Wall“, um Wände abspeichern zu können.
Die Typen „Stence“, „Wumpus“ und „Player“ sind nicht relevant für das Modellieren des Weltbildes. Das Erkennen des Wumpus wird auf andere Weise gelöst.
Der Typ „Start“ zeigt an, wo sich der Spieler ganz zu Beginn befand.

Damit der Spieler nicht auf Felder läuft, die ihn das Leben kosten oder überhaupt nicht begehbar sind, gibt es eine Methode „isWalkable“. Wenn der Typ „Wall“, „Hole“ oder „Breeze“ in der Menge der Typen enthalten ist, so gibt diese Methode „false“ zurück, d.h. dass dieses Feld nicht begehbar ist. „Breeze“ wird aus Unsicherheit auch als Unbegehbar angesehen.

Um nun ein Loch erkennen zu können, ist es interessant, was bereits für Typen in der Menge der Typen eines Feldes enthalten sind. Wenn bereits ein „Breeze“ enthalten ist, und nun wieder ein „Breeze“ gespürt wird, so wird der Menge der Typ „Hole“ hinzugefügt.
Wenn aber bereits ein „Empty“ erkannt wurde, so wird der Typ „Breeze“ nicht mehr mit aufgenommen. Wenn ein Feld nachträglich als „Empty“ erkannt wird, also das Feld indirekt als sicher empfunden wird, dann werden sowohl (falls vorhanden) „Breeze“ als auch „Hole“ aus dieser Menge entfernt.

Reines Erkunden

Zu Beginn wird das reine Erkunden ohne Erkennen des Wumpus erläutert. Hierbei ist beginnend die Klasse „MyWumpusAgent“ relevant. Um zunächst erstmal überhaupt Felder abspeichern zu können, gibt es eine Liste von Feldern namens „wumpusMap“. Diese Liste von Feldern liegt hauptsächlich unsortiert vor, da die Position eines Feldes in dem Feld selbst abgespeichert ist.
Zu Beginn ist bekannt, dass der Spieler auf der Position (1,1) in Richtung Osten startet. Da das Spielfeld rechteckig aufgebaut ist, kann man annehmen, dass links neben ihm und über ihm Wände verlaufen. Diese Wände verlaufen entlang der kompletten x- bzw. y-Linie. Daher werden die x- bzw. y-Koordinaten in Variablen abgespeichert. Wenn der Spieler sich nun bewegt und ein neues Feld auf der nördlichen oder östlichen Wandlinie liegt, so wird diesem Feld der Typ „Wall“ hinzugefügt. Somit liegen die möglichen begehbaren Felder direkt unter und rechts neben ihm. Wenn der Spieler nun anfangs bereits einen Luftzug spürt, so wird diesen beiden Feldern der Typ „Breeze“ hinzugefügt und das Spiel ist zu Ende, da „Breeze“-

Felder als nicht begehbar angesehen werden. Andernfalls sind beide Felder vom Typ „Empty“. Diese Felder werden direkt in der „wumpusMap“-Liste abgespeichert.

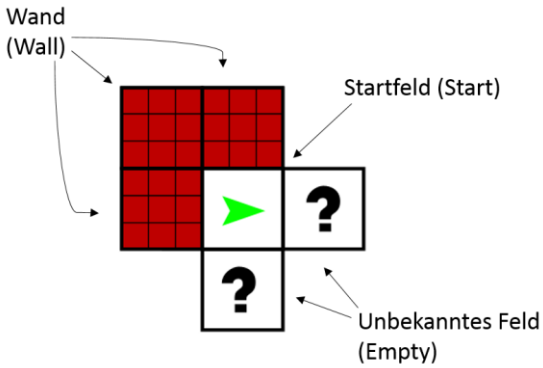

Abbildung 1: Startinitialiserung grafisch für Felder mit Typen „Empty“

Mit dieser Startinitialisierung versucht der Spieler nun die Welt zu erkunden. Hierzu sucht er immer das am naheliegendsten Feld vom Typ „Empty“. Um nun den Weg zu diesem Feld zu finden, gibt es die Klassen „FieldUtil“ und „TreeField“.

In der Klasse „FieldUtil“ gibt es die Methode „actionsToFieldType“. Diese erwartet als Parameter den Typen, welcher gefunden werden soll, eine Liste von Feldern, in der sich das zu suchende Feld befindet, den Startpunkt und die Anfangsrichtung, in welche der Spieler schaut. Bei der Suche wird der Weg zu dem Feld durch die HunterActions dargestellt. Hierzu wird ein leerer Stack namens „actionStack“ angelegt. Dieser leere Stack enthält später die HunterActions zu dem Feld mit dem zu suchenden Typen. Ebenso werden zwei neue Listen von Feldern angelegt. Die erste Liste (closedList) enthält später die Felder, welche bereits in der Suche besucht wurden. Die andere Liste (manhattenList) enthält Felder der Klasse „TreeField.
Ein „TreeField“ kennt sein Vorgängerfeld, die Richtung, in die der Spieler schauen würde, wenn er das Feld begeht bzw. später verlässt, und die Aktionen vom Vorgängerfeld zu diesem Feld als Stack gespeichert.
Die Manhattan-Distanz zu einem Feld ergibt sich also aus der Anzahl an Aktionen, die auf dem Stack gespeichert wurden. Die gesamte Suche über ist die „manhattenList“ genau nach dieser Anzahl an Aktionen aufsteigend sortiert. Damit wird sichergestellt, dass immer das naheliegendste Feld neu betrachtet wird und somit das naheliegendste Feld des zu suchenden Typs gefunden wird.

Bei der Suche werden nun nach und nach immer alle vier Himmelsrichtung betrachtet. Hierbei wird zunächst geschaut, ob das Feld überhaupt existiert, begehbar ist und noch nicht besucht wurde (den dann würde es einen kürzeren Weg dorthin geben). Wenn das Feld existiert, begehbar ist und noch nicht besucht wurde, so wird die Distanz anhand der Anzahl an Aktionen bestimmt, und anschließend wird das Feld der „manhattenList“ hinzugefügt. Wenn nun ein Feld mit dem gesuchten Typ gefunden wurde, so werden rückwärts alle Felder beginnend von dem gefundenen Feld mit dem gesuchten Typ abgearbeitet. Das Rückwärtsgehen ist möglich, da jedes Feld seinen Vorgänger kennt. Bei dem Rückwärtslaufen werden die Aktionen jedes einzelnen Feldes auf den „actionStack“ gelegt. Die oberste Aktion des „actionStack“ ist die Aktion, welche der Spieler im aktuellen Zug ausführt. Die Suche wird

nach jedem Zug neu gestartet, um auf später aufkommende Eventualitäten reagieren zu können.

Wenn nun der Spieler in die östliche oder südliche Wand läuft, so wird ebenfalls jeweils eine Variable mit der Koordinate gesetzt, auf welcher sich die Wand befindet. Durch den rechteckigen Aufbau der Karte, ziehen sich diese Wände ebenfalls komplett auf einer Linie durch. In Zukunft werden beim Erkunden der Karte auf dieser Linie liegende Felder direkt mit dem Typen „Wall“ versehen und sind somit als nicht begehbar markiert.

Wenn der Spieler beim Erkunden der Karte auf ein Feld mit Gold trifft, so bekommt dieses Feld den Typen „Gold“ zugewiesen. Das Aufsammeln muss nicht direkt von statten gehen, da der Wumpus sich in der Nähe befinden könnte.

Wumpus-Strategie

Um nun den Wumpus mit ins Spiel zu bringen, wurde zunächst eine Klasse „WumpusDangerMap“ angelegt. Der Hauptkern dieser Klasse ist die Methode „refreshMaps“, welche eine Liste von „WumpusFields“ namens „dangerFields“ mit möglichen Felder füllt, auf welcher sich der Wumpus befinden könnte. Der Unterschied zwischen einem „WumpusField“ und einem normalen „Field“ ist der, dass bei einem „WumpusField“ noch abgespeichert wird, welche Wumpi sich dort befinden könnten. Ein Wumpus wird hierbei durch die Klasse „WumpusBot“ repräsentiert.

Bei dem Befüllen der Liste „dangerFields“ mit möglichen Felder gibt es folgendes Vorgehen. Zunächst wird die zuletzt gerochene Intensität („lastIntensity“) auf einen Wert höher des maximalen Riechwertes gesetzt, um zu erkennen, ob der Wumpus überhaupt gerochen wurde. Danach wird ein sogenannter „Scope“ angelegt. Der Scope spiegelt den Bereich wider, in welchem sich ein Wumpus überhaupt befinden könnte, falls einer gerochen wird.

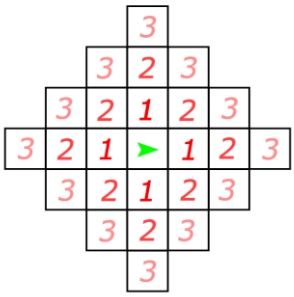

Abbildung 2: Scope mit allen möglichen Feldern

Dabei werden bereits die Felder ausgeschlossen, welche sich außerhalb des Spielfeldes befinden (Es kann leider nicht die Methode „isWalkable“ von der Klasse „Field“ verwendet werden, da es möglich ist, dass das Weltbild des Spielers nicht korrekt ist). Anschließend wird geschaut, ob sich ein Wumpus bewegt hat. Wenn sich ein Wumpus bewegt hat, so wird ein Zähler (globalRumbleCounter) hochgezählt. Dieser ist vor allem relevant, wenn ein Wumpus neu entdeckt wird.
Nun werden für jeden bereits entdeckten Wumpus ein paar Werte gesetzt. Falls es gerumpelt hat, wird zunächst für jeden Wumpus separat noch ein Zähler hochgezählt. Dieser Zähler enthält immer die mögliche Anzahl an Schritte, die ein Wumpus gemacht haben könnte, wenn

er sich nicht im aktuellen Scope befindet. Falls der Wumpus im aktuellen Zug nicht gerochen wurde, wird für den speziellen Wumpus die zuletzt gerochene Intensität auf „lastIntensity“ gesetzt (Diese beinhaltet bekanntlich aktuell einen Wert höher des maximalen Riechwertes). Ebenso wird der Wumpus als nicht gerochen markiert. Zuletzt werden jedem bereits gerochenem Wumpus mögliche Bewegungsaktionen hinzugefügt (siehe Methode „refreshActions“ in der Klasse „WumpusBot“). Wenn der Spieler ein rumpeln gehört hat, so werden alle möglichen Aktionen hinzugefügt. Das ist vor allem notwendig, wenn mehrere Wumpi vorhanden sind. Andernfalls werden lediglich die Aktionen „Turn-Left“, „Turn-Right“ und „Sit“ hinzugefügt.

Wenn der Wumpus nun im aktuellen stenchRadar ist, also sich im Scope befindet, wird geschaut, ob dieser das erste Mal gerochen wurde. Falls dies der Fall ist, wird ein neues Objekt vom Typ „WumpusBot“ angelegt, der aktuelle globale Zählerstand (globalRumbleCounter) des Rumpelns ihm übergeben, sowie die Riechdistanz und ob er sich zum aktuellen Zeitpunkt bewegt haben könnte. Das WumpusBot-Objekt wird in einer HashMap gespeichert und als Key wird die ID des Wumpus verwendet. Im Anschluss wird die Methode „refreshWumpus“ der Klasse „WumpusBot“ aufgerufen. In dieser Methode wird versucht, durch die reine Manhattan-Distanz, Felder, auf denen sich der Wumpus befinden könnte, auszuschließen (siehe Klasse „FieldUtil“ Methode „distance“ zum Berechnen der Manhattan-Distanz).
Hierbei wird beginnend geschaut, welche Felder auf Basis der gerochenen Intensität überhaupt relevant sind (In der folgenden Abbildung ist beispielhaft aufgeführt, welche Felder relevant sind, wenn die Intensität 2 beträgt).

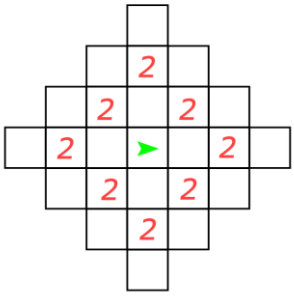

Abbildung 3: Scope mit Feldern der Intensität 2

Wenn der Wumpus das erste Mal gerochen wurde, so kann dieser sich auf jedem der gerochenen Felder befinden. Diese möglichen Felder werden wiederrum in einer Liste namens „tNewFields“ abgespeichert. Wenn der Wumpus erneut gerochen wird, so findet ein Vergleich der Distanzen statt. Und zwar wird von jedem vorher gerochenen Feld, welche sich in der Liste „dangerFields“ befinden, zu jedem aktuell gerochenen Feld betrachtet, ob die reine Manhattan-Distanz kleiner oder gleich der Anzahl an Schritten ist, die der Wumpus gemacht haben könnte. Wenn dies der Fall ist, so wird das neu gerochene Feld in die Liste „tNewFields“ aufgenommen, auf die sich der Wumpus befinden könnte. Am Ende wird die Liste „dangerFields“, welche die eigentliche Liste mit den möglichen Felder ist, auf denen sich der Wumpus befinden könnte, auf die neu gefundenen möglichen Felder gesetzt. Die Liste „dangerFields“ enthält somit bis zum nächsten Riechen die Felder, auf denen sich der Wumpus zuletzt befinden konnte.
Nachdem dies für jeden Wumpus individuell getan wurde, werden wieder in der Methode „refreshMaps“ der Klasse „WumpusDangerMap“ die noch unlogischen Felder entfernt, auf denen sich ein Wumpus nicht befinden kann. Zum Beispiel wird vom aktuellen Zeitpunkt

zurückgeschaut, ob sich der Wumpus bewegt hat. Wenn dies nicht der Fall ist, dann werden immer die Felder entfernt, auf denen sich der Spieler zeitweise befand. Oder wenn der Wumpus aktuell nicht mehr gerochen wird, vorher aber mit der Intensität 3 (also der am weitesten möglichen Entfernung) gerochen wurde, so werden alle Felder entfernt, die sich im aktuellen Scope befinden. Das ist möglich, da man ihn sonst ja riechen würde.

Wenn sich nun mehrere Wumpi auf der Karte befinden, so werden alle Felder, auf denen sich ein Wumpus befinden könnte, zu einer Liste zusammengefügt. Hierbei werden die jeweiligen „dangerFields"-Listen zu einer großen Liste zusammengesetzt. Diese Liste dient nun als Grundlage, ob der Spieler einen Schuss wagen soll, oder eher weglaufen soll.

Ein Schuss wird in mehreren Fällen durchgeführt. Zunächst gibt es die sogenannten „tryShots". Hierbei wird versucht ein Wumpus durch Glück zu treffen. Wenn die letzte Intensität 2 war, der Wumpus also recht nah ist, wird geschaut, ob der Wumpus sich auf einem Feld befinden könnte, in welche Richtung der Spieler gerade schaut. Ist dies der Fall wird ein Schuss abgefeuert. Dies passiert maximal 2 Mal, um nicht sinnlos alle Pfeile zu verschießen.
Der zweite Fall wo ein Schuss abgefeuert wird, ist wenn die letzte Intensität 1 war, der Wumpus also unmittelbar neben einem steht. Hierbei wird nun das erste Feld aus der Liste „dangerFields" genommen. Nun werden die Aktionen zu diesem Feld ermittelt. Dies geschieht ähnelnd der Suche nach einem Feld bestimmten Typs. Wenn nun die Anzahl an Aktionen kleiner 2 ist, also der Spieler auf das Feld schaut, wird eine Schuss abgefeuert.
Bei dem letzten Fall, wo ein Schuss abgefeuert wird, kann genau gesagt werden, dass ein Wumpus sich nur noch auf einem bestimmten Feld befindet. Dabei wird dann zu diesem Feld navigiert. Wenn der Spieler nun zu diesem Feld schaut, schießt er los. Das Navigieren zu einem Feld geschieht wieder ähnlich dem Suchen eines Feldes bestimmten Typs. Ist es nun zu unsicher einen Schuss bzw. einen Versuchs-Schuss abzufeuern, so läuft der Spieler vor dem Wumpus weg. Hierbei werden die unmittelbar anliegenden Felder angeschaut. Für jedes unmittelbar anliegende Feld wird die minimale Distanz zu jedem möglichen Feld, auf denen sich der Wumpus befinden könnte, ermittelt. Danach wird das Feld angegangen, das die größte Distanz zu allen Wumpus-Feldern hat, da dieses am sichersten ist (siehe Methode „findSafeField" in der Klasse „WumpusDangerMap"). Wenn der Spieler nun keine Pfeile mehr haben sollte, so wird immer das sicherste Feld gesucht.

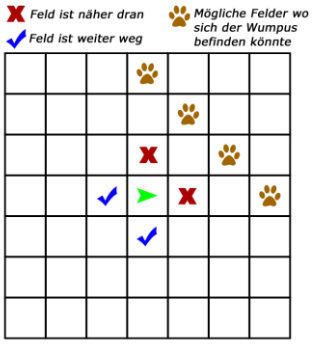

Abbildung 4: „findSafeField" grafisch dargestellt

Wenn ein Schuss abgefeuert wurde, so können zwei Fälle eintreten. Entweder wurde ein Wumpus getötet oder der Schuss traf keinen Wumpus. Bei dem Fall, dass der Wumpus nicht

getroffen wurde, wird das Feld entfernt, in welche Richtung der Schuss gegangen ist. In dem anderen Fall, wenn ein Wumpus getötet wurde, werden die „dangerFields“ zurückgesetzt.

In der Klasse „MyWumpusAgent“ wird in der Methode „action“ nun nacheinander geschaut, ob zunächst der Wumpus gefährlich nahe ist bzw. abgeschossen werden kann. Wenn dies nicht der Fall ist, so wird geschaut, ob das Gold bereits gefunden wurde. Wenn ja, dann laufe zurück zum Startpunkt. Durch die nun schon entstandene Reihenfolge wird bereits sichergestellt, dass der Spieler auf dem Rückweg nicht einfach stumpfsinnig in der Wumpus läuft. Falls keines dieser vorherigen Bedingungen eintritt, so wird geschaut, ob der Spieler bereits irgendwo ein Feld mit dem Typen „Gold“ gefunden hat. Wenn ja, dann navigiere zu dem Feld mit dem Typen „Gold“. Falls der Spieler auch noch kein Feld mit Gold gefunden hat, so suche die Welt weiter nach dem Gold ab. Wenn am Ende selbst die erstellte interne Karte keine begehbaren Felder mehr hergibt, dann laufe zum Startpunkt zurück.