Men's Morris - Trennung von Logik und Darstellung mit WPF und MVVM

Lukas Körfer

Aachen, am 31. August 2016

Seit 2015 veranstaltet IT-Talents unter dem Namen *Code Competition* Programmierwettbewerbe mit unterschiedlichsten Aufgaben. Die Aufgabenstellung von August 2016 stammt dabei von einem Partnerunternehmen, der Gauselmann AG. Diese sieht vor, das Brettspiel Mühle umzusetzen, inklusive einer grafischen Oberfläche sowie einer künstlichen Intelligenz, gegen die als menschlicher Spieler angetreten werden kann. Um diese Aufgabe qualitativ hochwertig zu bearbeiten, wird im Rahmen dieser Umsetzung mit dem Projektnamen *Men's Morris* auf eine strikte Trennung der einzelnen Komponenten (Spiel-Engine, Spiel, künstliche Intelligenz) geachtet. Als weiteren Schwerpunkt werde ich den Einsatz des MVVM-Schemas thematisieren, für dessen Verwendung das .NET beigefügte Grafik-Framework *Windows Presentation Foundation* (WPF) optimiert wurde. MVVM (Model-View-ViewModel) führt dazu eine Präsentationslogikebene ein, an deren Eigenschaften die Präsentationsebene binden kann, wodurch beide automatisiert synchronisiert werden.

Inhaltsverzeichnis

1 Programmaufbau

Die Spiel-Engine ist so aufgebaut, dass sie keinerlei Verbindung zur tatsächlichen Umsetzung des Spieles besitzt, um sie unter Umständen in einer Neuimplementierung des Spieles komplett wiederverwenden zu können. Sie definiert zum einen Schnittstellen, um das Spiel zu starten und über Änderungen zwecks Aktualisierung der Oberfläche informiert zu werden. Zum anderen bietet sie eine Spieler-Schnittstelle an, die das Erstellen von Bots, also computergesteuerten Spielern erlaubt. Interessierte Dritte könnten über die Implementierung dieser Schnittstelle Inhalte erzeugen, die dann als spielende Akteure in das Spiel aufgenommen werden.

Sowohl das Spiel als auch die einzelnen Bots können auf jegliche Eigenschaften jedes Spielelementes (Spielfeldpositionen, Spielsteinen, Spielern) zugreifen, diese zu verändern ist jedoch der Spiel-Engine vorbehalten. Selbst das Spiel, das eine Instanz der Spiel-Engine erzeugt und verwaltet, kann diese nicht manipulieren.

Eine Besonderheit der Spiel-Engine ist die Möglichkeit, im Spiel verschiedene Aktionen und Züge zu simulieren und dann wieder rückgängig zu machen. Diese Möglichkeit können Bots nutzen, um zusätzliche Informationen über die Güte von Zügen oder die Spielsituation nach verschiedenen Zügen zu erhalten.

Da die einzelnen Positionen des Spielfeldes und ihre Verbindungen nicht festgelegt sind, sondern beim Start einer Partie generiert werden, ist es möglich Parameter für die Generierung zu verwenden, die dann zur Erzeugung alternativer Spielfelder entsprechend modifiziert werden können. Durch diese Anpassbarkeit kann das implementierende Spiel auf einfache Weise Varianten des klassischen Mühle-Spiels anbieten, wie sie beispielsweise die englischsprachige Wikipedia[1] aufführt. Diese große Flexibilität führte letztendlich auch zum Namen *Men's Morris*, da neben dem geforderten **Nine** Men's Morris problemlos auch **Six** Men's Morris, **Twelve** Men's Morris und viele mehr gespielt werden können.

[1] https://en.wikipedia.org/wiki/Nine_Men's_Morris#Variants

2 Grafische Oberfläche

Die grafische Oberfläche des Spiel wird mit Hilfe von WPF verwirklicht. Obwohl WPF auch die direkte Ansteuerung der Elemente der grafischen Oberfläche ermöglicht, ist die Verwendung des MVVM-Schemas und Datenbindung oft die bessere Wahl. Durch sogenannte ViewModels können die Objekte des Modells angesteuert werden und verwalten die notwendige Präsentationslogik. So kennen beispielsweise die einzelnen Positionen des Mühle-Spielfeldes ihre Positionen nicht, ihnen wird dann ein ViewModel zugewiesen, das solche Informationen ermittelt. Die tatsächliche Ansicht (View) sorgt dann für eine Positionierung an der ermittelten Stelle.

Analog zu den konkreten Elementen eines Mühle-Spiels wie den Spielsteinen, den Positionen oder den Positionsverbindungen werden auch die abstrakten Spielelemente wie mögliche Züge durch ViewModel und zugehörige Views abgebildet. Alle genannten ViewModel werden gemeinsam an die Spielfeld-Ansicht gebunden und WPF ermittelt selbstständig, welche Ansicht für welches Ansichtsmodell gewählt und dann gezeichnet wird.

Entsprechend der soeben erläuterten Aufteilung in Modell, Ansicht und Ansichtsmodell erfolgt auch die Unterteilung innerhalb des Projektes in die Ordner *View* und *ViewModel*, da die einzelnen Modelle aus der Spiel-Engine stammen. Der Ordner *Helpers* beinhaltet Konvertierungsfunktionen, die auf einfache Weise in der WPF-Auszeichnungssprache XAML verwendet werden können. Weitere Elemente des Spiel-Projektes sind das Hauptfenster, in dem die einzelnen Ansichten dann entsprechend ihrer Funktionalität verwendet werden, sowie kleinere Programmteile, die für eine Bereitstellung der einzelnen Computerspieler (Bots) oder der voreingestellten Spielvariationen sorgen.

Die resultierende Benutzeroberfläche besteht aus zwei Ansichten. Die erste dieser Ansichten ist die Startansicht, in der der Nutzer Einstellungen über die antretenden Spieler und die zu spielende Mühle-Variation (siehe Kapitel 1) vornehmen kann. Es ist zudem möglich, durch die Modifikation einiger Parameter unter der Spielauswahl - *Free Choice* - eine eigene Mühle-Variation zu kreieren. In der zweiten Ansicht findet das eigentliche Spiel statt. Dabei wird auf der linken Seite neben der Möglichkeit, zur Startansicht zurückzukehren, angezeigt, welcher Spieler (rot oder blau) derzeit am Zug ist. Auf der rechten Seite befindet sich das Spielfeld, auf dem die einzelnen Spielelemente sowie mögliche auswählbare Aktionen für menschliche Spieler aufgezeichnet sind. Sollte das Spiel beendet sein, wird der Sieger des Spiels hervorgehoben und es werden dem Nutzer die Möglichkeiten präsentiert, das Spiel in der gleichen Konfiguration erneut zu starten oder zwecks Anpassung der Spielparameter in die Startansicht zurückzugehen.

3 Künstliche Intelligenz

Beschäftigt man sich mit der Idee, einen Computerspieler oder Bot für das klassische Mühle-Spiel zu entwickeln, so fällt zunächst auf, dass das Spiel im mathematischen Sinn vollständig gelöst ist. Das bedeutet, dass Datenpakete existieren, die jede mögliche Spielsituation und den dafür besten Zug beinhalten. Da diese Daten jedoch einen erheblichen Umfang haben, ist es unmöglich ein Computerprogramm mit diesem Wissen auszustatten und dann auszuliefern.

Ein weiterer Ansatz, um dem zu erzeugendem Computerspieler eine künstliche Intelligenz zuzuführen, ist die Simulation möglicher Spielzüge und die Ermittlung des besten dieser Züge (auf Basis einer Bewertung wie der Anzahl der Steine des eigenen Spielers). Da die Simulation vieler verschiedener Züge und ihrer daraufhin auf beiden Seiten erfolgenden Gegenzüge jedoch sehr zeitintensiv ist, soll hier eine dritte Methode umgesetzt werden.

Es handelt sich bei dieser Methode um die Bewertung eines jeden möglichen Zuges in einer Runde durch eine Heuristik[2]. Dabei wird für jeden möglichen Zug das aus diesem Zug resultierende Spielfeld betrachtet, um dann verschiedene Eigenschaften dieses Spielzustandes zu ermitteln. Die Werte dieser Eigenschaften ergeben gemeinsam mit Gewichtungskoeffizienten eine Evaluierungsfunktion, die dann den besten möglichen Zug ermittelt.

[2] http://www.dasconference.ro/papers/2008/B7.pdf