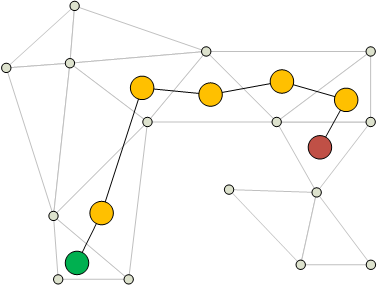
Bewegte Objekte lassen sich durch Transformationen bewegen. Hierfür werden vom bewegenden Objekt mindestens zwei Instanzen benötigt und mittels Tweening animiert. Ein anderer Ansatz ist eine Bewegung innerhalb eines vorgegebenen Pfades. Pfade benötigen im Gegensatz zum Tweening nur eine Objektinstanz. Pfade haben den Vorteil effizienter zu sein, da sie weniger internen Verwaltungsaufwand besitzen. Damit sind sie besser geeignet um Objekte auf einem vorgegebenen Weg zu führen. Pfade können innerhalb von Karten (Map) durch einen Algorithmus (z.B. schnellster oder kürzester Weg), automatisiert erzeugt werden. Diese Pfade werden Routen (route) genannt und können verwendet werden, um ein Objekt (z.B. Spielfigur) zu einem Punkt (z.B. bei Mausklicks) zu führen.



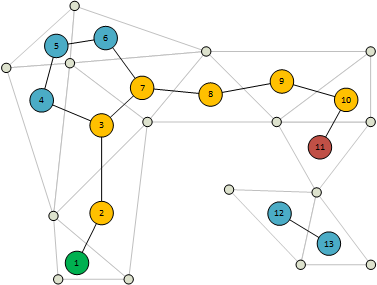
*Abb* 1: Pfad innerhalb einer Karte

Die Karte besteht aus (mindestens drei) Scheitelpunkte (Vertex), welche ein Netz von Dreiecksmaschen (Mesh) aufspannen. An jeden Scheitelpunkt sind Eigenschaften gebunden, welche das Aussehen von Objektinstanzen manipulieren. Diese Eigenschaften haben zum Beispiel Auswirkungen auf die Rotation und Skalierung. Die Dreiecksmaschen interpolieren diese Eigenschaften, so dass für jeden Punkt (MapPoint) auf der Karte individuelle Eigenschaften gelten.

Ein Pfad/ Route besteht aus einem Start- (StartingPoint), einem End- (EndPoint) und einer variablen Anzahl von Wegpunkten (WayPoint). Ein Pfad/ Route hat eine Ablaufrichtung, welche vom Start über die Wegpunkte zum Endpunkt führt. Für Pfade gilt zudem, wenn der Start- und Endpunkt gleich ist, handelt es sich um einen Rundpfad, welcher endlos abgelaufen werden kann. Für Pfade mit unterschiedlichen Start- und Endpunkten kann die Laufrichtung am End- und Startpunkt gewechselt werden, um hier ebenfalls eine endlose Animation zu ermöglichen. Jeder Wegabschnitt (WaySection) enthält zu jeder Laufrichtung einen Objektzustand, so dass Spielfiguren in der korrekten Ansicht gezeichnet werden können.

# Der Routingalgorithmus

Das Ermitteln einer Route erfolgt mit Hilfe eines Backtracking-Algorithmus.



*Abb* 2: Beispiel

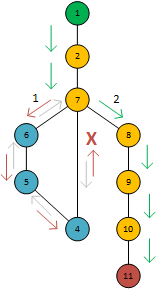
Aus dem Dreiecksmaschen-Netz wird ein Graph erzeugt. Dreiecke, welche eine gemeinsame Kante besitzen, werden im Graph miteinander verbunden. Es besteht zwischen den Knoten eine transitive Beziehung. Das heißt, wenn Knoten *a* mit Knoten *b* in Verbindung steht und Knoten *b* mit Knoten *c*, dann steht auch Knoten *a* in Verbindung mit Knoten *c*. Die Kanten des Graphen können zudem gewichtet und bei der Wegfindung berücksichtigt werden.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1 | x | k |  |  |  |  |  |  |  |  |  |  |  |
| 2 | k | x | k |  |  |  |  |  |  |  |  |  |  |
| 3 |  | k | x | k |  |  |  |  |  |  |  |  |  |
| 4 |  |  | k | x | k |  | k |  |  |  |  |  |  |
| 5 |  |  |  | k | x | k |  |  |  |  |  |  |  |
| 6 |  |  |  |  | k | x | k |  |  |  |  |  |  |
| 7 |  |  | k |  |  | k | x | k |  |  |  |  |  |
| 8 |  |  |  |  |  |  | k | x | k |  |  |  |  |
| 9 |  |  |  |  |  |  |  | k | x | k |  |  |  |
| 10 |  |  |  |  |  |  |  |  | k | x | k |  |  |
| 11 |  |  |  |  |  |  |  |  |  | k | x |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |  | x | k |
| 13 |  |  |  |  |  |  |  |  |  |  |  | k | x |

x = Reflektiv

k = gemeinsame Kante

Der Backtrackingalgorithmus arbeitet nach dem Versuch-und-Irrtum-Prinzip. Ausgehend vom Startpunkt werden die in Verbindung stehenden Knoten ermittelt und rekursiv zu den ersten gewechselt. In diesem werden wieder alle in direkter Verbindung stehenden Nachfolgeknoten ermittelt. Knoten, die bereits besucht wurden, werden ausgeschlossen. Wird eine Sackgasse oder das Ende erkannt, so wird die Rekursion beendet und zum vorherigen Knoten gewechselt und mit den nächsten Knoten fortgefahren. Mit diesem Algorithmus werden alle möglichen Wege gefunden. Durch die Summer der Gewichtungen kann ein Pfad aus der Menge der möglichen Pfade ausgewählt werden.



*Abb* 3: Backtrackingalgorithmus