Flow Tiles

Albert-Ludwigs-Universität Freiburg
Grafische Datenverarbeitung
Seminar Computer Animation WS 05/06
Dozent: Prof. Dr. Mathias Teschner
Referent: Sanja Jahnke
19.01.2006

Inhaltsangabe

0. Motivation

Flow- Tiles ist eine Methode zum Entwurf und zur Darstellung 2dimensionaler Geschwindigkeitsfelder, welche die Bewegungen vieler natürlicher Prozesse – wie den Fluss von Flüssigkeiten oder Menschenmengen – lenken.

Jedes Flow- Tile (deutsch: Fluss-Kachel) enthält ein kleines Feld, das bei einmaliger Speicherung mehrmals benutzt werden kann. Auf diese Weise bieten Flow- Tiles eine Möglichkeit Speicherplatz zu sparen. Durch das Zusammenfügen vieler Tiles entstehen dann große Flows – die Tilings. Anders als bei anderen prozeduralen Flow- Generatoren gibt es hier eine Benutzerschnittstelle für den Entwurf der Tiles und des gesamten Flow- Feldes.

Bei der Konstruktion der Tilings ist es möglich eine Vielzahl von internen und externen Grenzkonditionen (Boundary Conditions) festzulegen und zu erfüllen. Dadurch können sie in größere Umgebungen eingefügt werden. Außerdem sind die Ecken und Kanten der Tiles so konstruiert, dass die Kontinuität des Flows auch über die Grenzen der Tiles – sprich: die Fugen zwischen den Kacheln – hinweg sichergestellt ist. Zusätzlich sind alle Tiles und auch die daraus entstehenden Tilings abweichungsfrei, d.h. die Masse des Flows bleibt überall erhalten. So können mit Flow- Tiles eine Vielzahl von Effekten dargestellt werden, wie z.B. ein Fluss, der Blätter mit sich führt, eine Menschenmenge, die sich auf den Straßen einer Stadt bewegt, und wirbelnder Nebel. Diese drei Anwendungen werden im Folgenden erläutert, ebenso wie der Entwurf der Tiles, die Algorithmen zum Erstellen von Tilings und deren Einfügung in (interaktive) Umgebungen.

1. Einführung

Flows sind ein belebter und wichtiger Teil vieler Animationen, da sie die Bewegung unzähliger natürlicher Phänomene lenken.

Jedes Flow- Tile definiert eine kleine, stationäre Region des Geschwindigkeitsfeldes. Mehrere Tiles können zu einem großen, stationären Feld zusammengefügt werden, welches dann Flüssigkeiten, Menschenmengen oder andere Effekte leitet (s. Abb.1). Das Zusammenfügen der Tiles übernimmt der Benutzer, der dabei von einer Benutzerschnittstelle unterstützt wird. So kann auch ein ungeübter Benutzer stimmige, abweichungsfreie Flows erstellen.

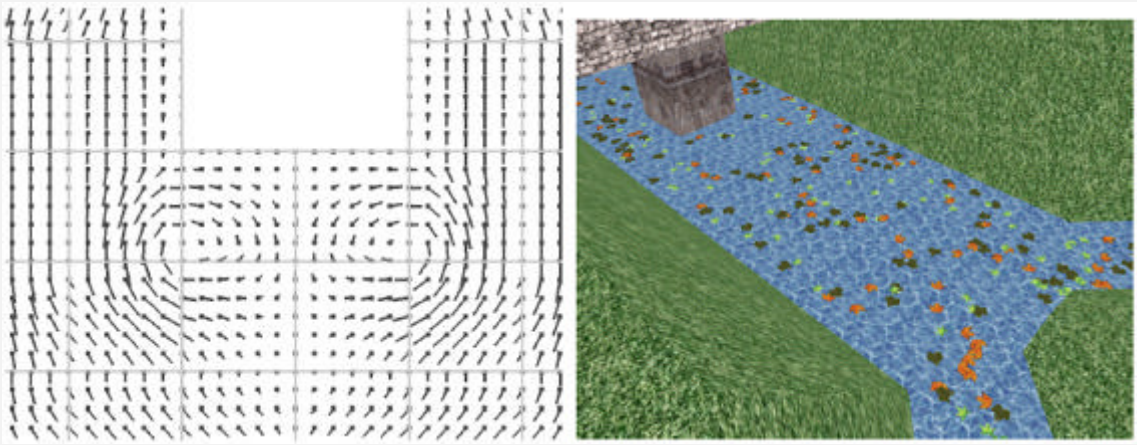


Abbildung 1: Flow- Tiles definieren einen Fluss. Jedes der Tiles im linken Raster enthält ein kleines abweichungsfreies Geschwindigkeitsfeld. Die Grenzkonditionen wurden benutzt, um den Flow um den Brückenpfeiler herum zu lenken. Rechts sieht man den Flow eingebettet in eine Anwendung, in der er die Blätter auf der Flussoberfläche führt.

Bei dem Entwurf von Flows ergeben sich im Allgemeinen zwei bezeichnende Probleme: wenn spezielle Fluss-Merkmale gewünscht sind, sind sie schwer zu gestalten, und bei Gebrauch von detaillierter Bewegung ist der Umfang von Effekten, die generiert werden können, begrenzt. Mit Flow- Tiles lassen sich diese Probleme handhaben, da sie die Möglichkeit bieten die Details des Flows selbst zu entwerfen und die Vielfalt der Effekte unterstützen. Denn durch die Verwendung eines kleinen Sets von prototypischen Tiles, die beliebig oft benutzt werden können, wird der Entwurf vereinfacht, da es nicht nur einfach ist die Tiles zu etwas größerem zusammenzufügen, sondern da das Ergebnis auch offensichtlich ist und nicht von *Blending*, Interpolation oder anderen intertile'schen Effekten abhängt. Außerdem ist diese Methode effizient, denn die Daten, die jedes einzelne Tile enthält, müssen nur einmal gespeichert werden, egal wie oft das jeweilige Tile verwendet wird.

Wie bei jedem Tile- Set müssen die Tiles (Kacheln) bestimmte Anforderungen erfüllen, damit sie so verlegt werden können, dass man zwischen ihnen keine Fugen erkennen kann. Deshalb sind die Ecken und Kanten der Flow- Tiles mit Kontinuitätsbedingungen belegt. Außerdem obliegen die Flow- Tiles Bedingungen zur Masseerhaltung und Grenzkonditionen (Boundary Conditions), denn für viele Anwendungen muss alles, was in ein Tile hineinfließt auch wieder heraus fließen. In Abschnitt 2 werden diese Anforderungen an Flow- Tiles, der Entwurf eines Flow- Tile- Sets und des Geschwindigkeitsfeldes innerhalb eines Tiles weiter erläutert.

Durch die Konditionen der Umgebung, in die der Flow eingefügt werden soll, entstehen die Grenzkonditionen, die der Flow erfüllen muss, z.B. bleibt ein Fluss innerhalb seines Flussbettes, Menschen laufen nicht durch Wände, usw. Aus diesen Grenzkonditionen des Tilings ergeben sich Einschränkungen auf die Platzierung jedes einzelnen Tiles innerhalb des Feldes, nicht nur auf die Tiles direkt an einer Grenze. Wenn z.B. ein Masseerhaltender Flow eine Sackgasse füllen soll, muss der Netto-Fluss am Eingang der Sackgasse Null sein, auch wenn innerhalb der Gasse einige Tiles platziert werden. Andernfalls müssten Menschen innerhalb der Gasse verschwinden oder auftauchen, was eine Abweichung der Masseerhaltung darstellen würde.

Flow- Tiles unterstützen diese internen und externen Grenzkonditionen des Tilings, damit es in Umgebungen und Anwendungen eingefügt werden kann. Die Bedingungen des Platzierens der Tiles und Algorithmen zum Erstellen des Tilings werden in Teil 3 weiter ausgeführt.

Flow- Tiles opfert die physikalische Plausibilität anderer Methoden – wie Simulation –, um die direkte Kontrolle über das Tiling- Ergebnis zu erhalten und Speicher- und Berechnungskapazität zu sparen. Denn bei Flow- Tiles obliegt es dem Benutzer plausible Flows zu erzeugen. Die Tiles und Tiling- Algorithmen enthalten nur Bedingungen, die helfen die wichtigsten physikalischen Anforderungen zu erfüllen, wie z.B. zu verhindern, dass man eine Menschenmenge in eine Sackgasse führt.

Flow- Tiles können hauptsachlich in Echtzeit-Anwendungen eingesetzt werden, bei denen umfangsreiche Flows auf ebenen Flächen (wie Straßen, Wasseroberflächen, etc.) benötigt werden, um Animationen zu lenken. In Teil 5 werden zwei Beispiele für stationäre Flows vorgestellt, bei denen sich das Feld selbst über die Zeit hinweg nicht verändert: im ersten führen Flow- Tiles Menschen durch die Straßen einer Stadt, im zweiten fließt ein Fluss um Hindernisse herum. Außerdem werden auch zeit-veränderliche Flows und ihre Anwendung bei wirbelndem Nebel dargestellt.

2. Entwurf der Flow Tiles

Um mit Flow- Tiles einen großen Flow darstellen zu können, muss zuerst eine Menge kleiner Flow- Felder – die Tiles – erstellt werden, welche dann zu einem Tiling kombiniert werden. Das Paper von Steven Chenney beschränkt sich dabei auf 2dimensionale, quadratische Tiles. Heutzutage sind zwar fast alle Animationen 3D, und die Tiles könnten auch dementsprechend erweitert werden, aber für die Anwendungen, die in diesem Paper beschrieben werden, reichen die flachen 2D Flows aus. Außerdem können die fertigen Tilings beim Einbinden in ihre Umgebungen (erläutert in Teil 4) so verzerrt werden, dass sie 3dimensional wirken (siehe Abbildung 1). Die quadratische Form der Tiles schränkt die Formen der Gebiete ein, die damit verkleidet werden können (mehr dazu in Teil 3 und 4), doch sie vereinfacht auch die Platzierung der Tiles, ihre Darstellung und Berechnung.

Die Flow- Tiles müssen abweichungsfrei sein und die Masse erhalten, damit die späteren Anwendungen auch realistisch sind – und nicht etwa bei dem Beispiel der Menschen auf Straßen fünf Personen ein Tile betreten, es aber nur eine wieder verlässt, und der Rest irgendwo verloren geht –. Zusätzlich müssen die Tilings spezielle Grenzkonditionen erfüllen, damit sie in größere Umgebungen eingefügt werden können. Abweichungsfreiheit und Grenzkonditionen haben Folgen für die Darstellung und den Entwurf der Tiles, die im Folgenden erläutert werden.

Außerdem sind Flow- Tiles stationär, d.h. sie verändern sich nicht über die Zeit, aber durch die zeitliche Kombination von Tilings können dynamische Flows kreiert werden.

2.1 Flow- Tiles- Sets

Bei Flow- Tiles wird mit einer endlichen Menge von Tiles gearbeitet, um den Speicherbedarf einzuschränken und die Benutzerschnittstelle zu vereinfachen. Diese Menge von Tiles bildet das Tile- Set, aus welchem das Tiling zusammengefügt wird. Zur effizienten Konstruktion des Tilings bedarf es einen Index, der die jeweiligen Tiles eindeutig identifiziert, denn die Tiling- Algorithmen bestimmen zuerst Einschränkungen auf das Tile, welches ins Tiling eingefügt werden kann, und suchen dann im Tile- Set nach dementsprechenden Tiles.

Es gibt verschiedene Methoden den Tile- Index festzulegen. Die eine indiziert die Tiles mit ihren Kantenwerten, die andere benutzt nur die Werte der Eckpunkte, da bei ihr für jede Kombination der Eckpunkte nur eine mögliche Kante existiert. Eine weitere verwendet Kanten- und Eckpunktwerte, um einen eindeutigen Index zu erzeugen. Diese letztere Variante wird auch bei den Flow- Tiles benutzt.

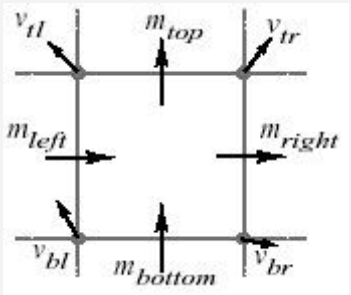


Abbildung 2: Tile Flow Methode zur Indizierung der Tiles.

Der Flow- Tile- Index besteht folglich aus acht Zahlen: den vier Geschwindigkeiten an den Eckpunkten der Tiles und dem Netto- Flow – dem ***Flux*** – über jede der vier Kanten (siehe Abbildung 2). In der *Fluids- Literatur* wird der Flux auch als Maßeinheit des Flow- Volumens bezeichnet.

Die Ecken werden betrachtet, weil sie den Ort darstellen, an welchem sich vier Tiles überlappen, und weil sie benutzt werden können, um die Kanten zu bestimmen. Der Flux über die Kanten wird im Index gebraucht, da er essentiell für die Kontrolle der Abweichungsfreiheit ist.

Jeder Eckindex v_i ist eine positive Zahl, die einen Array von möglichen Eckgeschwindigkeiten indiziert. Somit wird nur die Geschwindigkeit, aber nicht die Richtung des Flusses definiert. Die Menge der Eckgeschwindigkeiten, die in einem Tile- Set verwendet werden, legt der Benutzer fest. Dabei gibt es keinerlei Einschränkungen, die v_i müssen

lediglich positiv sein, und es ist sehr nützlich die Null-Geschwindigkeit mit einzuschließen um die Grenzbedingungen erfüllen zu können.

Die Kantenindexe m_i werden mit einer Basis- Flux- Einheit f_{base} multipliziert, um den tatsächlichen Flux über die jeweiligen Kanten zu erhalten. Dieser tatsächliche Flux ist definiert als Integral über den senkrecht zur Kante stehenden Anteil der Geschwindigkeitsvektoren an jedem Punkt der Kante (s. Abb.3). Der zur Kante parallele Anteil der Vektoren hat keinen Einfluss auf ihn. In den Diagrammen wird der Flux vereinfacht durch einen senkrecht zur Tile- Kante stehenden Pfeil dargestellt, doch dies sollte nicht vergessen lassen, dass er von der Geschwindigkeit an jedem Punkt der Kante abhängt. Nun gibt es aber entlang einer Kante unendlich viele Geschwindigkeitsfelder, die die gleichen Eckgeschwindigkeiten haben und zum gleichen Flux integrieren. Man könnte einen zusätzlichen Kantenindex zur Spezifizierung des Geschwindigkeitsfeldes einführen, aber hier wird nur ein einheitliches Feld benutzt, definiert durch die Eckgeschwindigkeiten und den Flux.

Abbildung 3: Flux an einem Punkt der Tile- Kante.

Für Flow- Tiles gilt die Konvention, dass die rechten und oberen Kanten- Fluxe positiv sind, und die unteren und linken negativ (s. Abb.2). Der Grund hierfür liegt darin, dass nur Tiles mit übereinstimmenden Ecken und Kanten nebeneinander platziert werden sollen, damit ein kontinuierlicher Flow auch über Tile- Grenzen hinweg entsteht. Dies wird durch die Konvention erleichtert, wie Abb.4 zeigt: Bei Tile A fließt der Flow über die obere Kante in das Tile und über die rechte Kante wieder heraus, d.h. $m_{A,top}$ = -2, $m_{A,right}$ = 2, $m_{A,left} = m_{A,bottom} = 0$. Bei Tile B fließt er über die linke Kante hinein und über die recht wieder heraus, folglich ist $m_{B,left}$ = 2, $m_{B,right}$ = 2, $m_{B,top}$ = 0, $m_{B,bottom}$ = 0. Damit nun Tile A neben Tile B platziert werden darf muss die Bedingung $m_{A,right} = m_{B,left}$ erfüllt sein, so wie in diesem Beispiel (mehr zum Zusammenfügen der Tiles in Abschnitt 3).

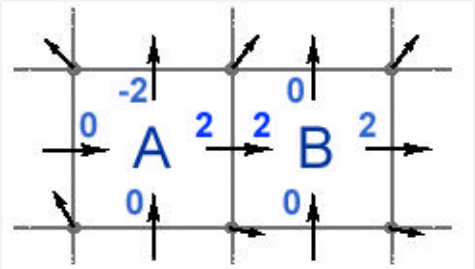


Abbildung4: Zwei Tiles mit übereinstimmenden Ecken und Kanten nebeneinander platziert.

Die höchsten und niedrigsten Werte, die der Flux in einem Tile- Set annehmen darf, legt der Benutzer fest, ebenso wie den Basis- Flux. Dabei gibt es jeweils einen m_{minx}- und $m_{max,x}$- Wert für die Menge der waagrechten Kanten (die oberen und unteren Kanten der Tiles) und einen m_{miny}- und $m_{max,y}$- Wert für die Menge der senkrechten Kanten (recht und links). Auf diese Weise wird sichergestellt, dass im Set keine Tiles entstehen, für die keine möglichen Nachbarn existieren. Würde man beispielsweise einen Höchstwert von 3 für m_{left} erlauben, aber maximal 2 für m_{right}, gäbe es keine Tiles im Set, welche die Gleichung $m_{right} = m_{left} = 3$ erfüllen.

Das Flow- Tile- Definitionsschema, bei dem sich der tatsächliche Flux aus dem Kantenindex multipliziert mit einer Basiseinheit ergibt (d.h. $f_i = m_i * f_{base}$), entstand aus dem Wunsch nach abweichungsfreien Tiles. Diese haben die für viele Anwendungen wichtige Eigenschaft, dass alles was in sie hineinfließt auch wieder heraus fließt, und umgekehrt. Mathematisch umgesetzt bedeutet diese Abweichungsfreiheit:

$$f_{left} + f_{bottom} = f_{right} + f_{top} \qquad (1)$$

Dabei ist f_{left} der Flux über die linke Tile- Kante, usw. Bei dem Entwurf der Tiles müssen folglich Kanten- Fluxe gewählt werden, die diese Bedingung erfüllen.

Zusätzlich möchte man die Möglichkeit Tiles zu kombinieren maximieren, d.h. man möchte für gegebene Eckgeschwindigkeiten und Kanten- Fluxe möglichst viele Tiles im Set haben, die diesen entsprechen. Deshalb lässt man für die Kantenindexe m_i nur ganzzahlige Werte zu. Der Benutzer legt für das jeweilige Tile- Set den Basis- Flux f_{base} und die

minimalen und maximalen Werte für die Kantenindexe m_{minx}, $m_{max,x}$, m_{miny} und $m_{max,y}$ fest. Dadurch wird gleichzeitig sichergestellt, dass ein endliches Tile- Set entsteht.

Nun bestehen die folgenden Gleichungen, die jedes Tile des Sets erfüllen muss:

$$\begin{aligned} f_{left} &= m_{left} f_{base} & m_{x,min} \le m_{left} \le m_{x,max} \\ f_{right} &= m_{right} f_{base} & m_{x,min} \le m_{right} \le m_{x,max} \\ f_{top} &= m_{top} f_{base} & m_{y,min} \le m_{top} \le m_{y,max} \\ f_{bottom} &= m_{bottom} f_{base} & m_{y,min} \le m_{bottom} \le m_{y,max} \end{aligned}$$

Da die jeweiligen Kanten- Fluxe aus den Kantenindexen multipliziert mit dem Basis-Flux berechnet werden, kann man diese Gleichungen in (1) einsetzen und f_{base} herauskürzen. Dies macht es möglich für die Abweichungsfreiheit nur die Kantenindexe zu benutzen:

$$m_{left} + m_{bottom} = m_{right} + m_{top} \qquad (2)$$

2.2 Speicherkosten des Sets

Die Anzahl der Flow- Tiles in einem vollständigen Tile- Set, welches aus allen möglichen Kombinationen der vom Benutzer festgelegten Werte für Kanten und Ecken resultiert, ist wichtig um die Speicherkosten des Sets einzuschätzen. Im Paper von Chenney ist eine Formel angegeben, mit der man diese berechnen kann, allerdings wurde der Beweis (aus Platzgründen) weggelassen, und die Ergebnisse der angegebenen Beispiele stimmen nicht direkt mit der Formel überein, da vermutlich ein konstanter Multiplikator fehlt. Die Anzahl der Flow- Tiles in einem vollständigen Tile- Set wächst (laut Paper) mit:

$$c^4 (M_x + M_y)^3$$

wobei $\boldsymbol{c}$ die Anzahl der möglichen Eckgeschwindigkeiten ist, und $\boldsymbol{M_x = m_{x,max} - m_{x,min}}$ und $\boldsymbol{M_y = m_{y,max} - m_{y,min}}$ die Anzahl der möglichen Fluxe über die horizontalen und vertikalen Kanten. Tile- Sets werden nun als $\boldsymbol{(M_x, M_y, c)}$ Tripel dargestellt und die Anzahl der Tiles berechnet. Wenn beispielsweise nur eine Eckgeschwindigkeit und drei Flux- Möglichkeiten zugelassen wurden, besteht das resultierende vollständige (3,3,1) Tile- Set aus 19 Tiles. Nach obiger Formel wären es aber $1^4 (3+3)^3 = 216$. Folglich bietet es sich an, das Ergebnis mit 19/216 zu multiplizieren, um das Ergebnis von 19 zu erhalten. Ein (3,3,5) Set enthält laut Paper 11875 Tiles [$5^4 (3+3)^3 = 135000$; $135000 * 19/216 = 11875$]. An diesen Beispielen sieht man wie schnell ein Set wächst, wenn man nur ein paar zusätzliche Möglichkeiten für die Eckgeschwindigkeiten oder die Kanten- Fluxe zulässt.

Die Praxis hat jedoch gezeigt, dass man auch mit einfachen Tile- Sets, die nur wenige Flux- und Geschwindigkeitsoptionen beinhalten, sehr komplexe Flows bilden kann. Das Ergebnis hängt mehr der Kombination der Tiles ab, als von der Komplexität der Tiles selbst, ähnlich wie aus einfachen, farbigen Kacheln ein komplexes Mosaik entstehen kann.

Außerdem fand man heraus, dass nur ein kleiner Teil des vollständigen Tile- Sets in den Anwendungen verwendet wird. Die potentiell große Menge von Tiles wird nur zu Beginn gebraucht, um dem Benutzer beim Auslegen der Tiles eine möglichst freie Wahl zu bieten, aber die unbenutzten Tiles müssen weder gefüllt (s. Abschnitt 2.3) noch gespeichert werden.

Zusätzlich verstärken Grenzkonditionen und die Bedingungen, denen das Auslegen der Tiles unterliegt, die Wiederverwendung einer kleinen Teilmenge des Tile- Sets, was uns wiederum davon befreit alle möglichen Tiles zu kreieren und speichern.

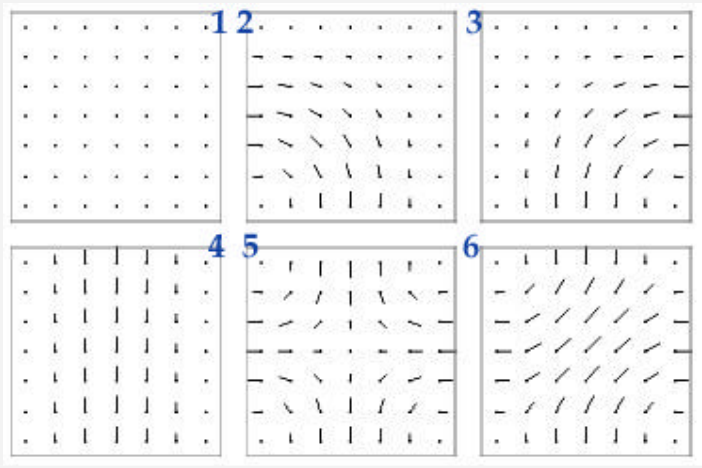
Der Fluss auf Abbildung 1 wurde beispielsweise aus nur 21 verschiedenen Tiles aus einer potentiellen Menge von 10tausenden Tiles gebaut.

Wenn aber trotzdem eine größere Menge von Tiles benötigt wird, kann die rotationale Symmetrie ausgenutzt werden, um Speicherkosten zu sparen. So kann ein einzelnes gespeichertes Tile durch Drehung um seine Achse bis zu vier Tiles repräsentieren, je nach Aussehen des Tiles.

Um dies an einem Beispiel zu erläutern: Wenn man die Symmetrie ausnutzt, nur eine Eckgeschwindigkeit von Null erlaubt und einen Flux von 1, 0 oder -1, bleiben vom (3,3,1) Tile- Set nur noch die sechs Tiles auf Abb.5 übrig:

Abbildung 5:

(3,3,1) Set unter Ausnutzung der rotationalen Symmetrie. Die einzig mögliche Eckgeschwindigkeit ist Null, der Flux 1, 0 oder -1.
Die Punkte markieren den Zellen-Mittelpunkt des Gitternetzes (in das jedes Tile aufgeteilt wurde), und die Linien, die von jedem Punkt ausgehen, die Richtung und Geschwindigkeit des Flusses innerhalb des Tiles.



Die Fluxe (bottom, right, top, left) der Tiles 1 bis 6 sind: (0,0,0,0), (1,0,0,-1), (1,1,0,0), (1,0,1,0), (1,1,-1,-1) und (1,1,1,1).
Die Tiles 2, 3, 4 und 6 repräsentieren jeweils vier Tiles, die durch Drehung um jeweils 90° entstehen. Tile 5 repräsentiert nur zwei Tiles, da es nach einer Drehung um 180° wieder in sich selbst übergeht, und Tile 1 bleibt immer dasselbe. So erhält man wegen den Symmetrie-Eigenschaften aus den hier abgebildeten sechs Tiles wieder die vollständigen 19.

2.3 Füllen der Flow- Tiles (Strömungsfunktion)

Nachdem die Eckgeschwindigkeiten und Kanten- Fluxe der Flow- Tiles festgelegt wurden, muss nun jedes einzelne Tile mit einem kontinuierlichen Flow- Feld gefüllt werden. Um dieses Feld zu speichern wird über jedes Tile ein nach den Achsen ausgerichtetes Gitternetz (Raster) gelegt (siehe Abb.6). Jede Kante dieses Gitters speichert an ihrem Mittelpunkt die Geschwindigkeit senkrecht zur Kante. Diese Methode erfordert bei einem 2D Gitter mit nxn Zellen die Speicherung von 2n(n+1) Werten. Doch die Abweichungs-freiheit des Feldes und die damit verbundenen Bedingungen ermöglichen es die Speicherkosten zu halbieren.

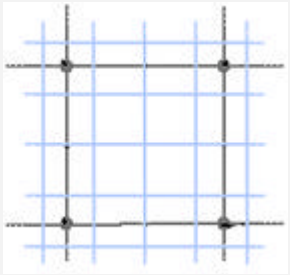


Abbildung 6: Tile mit Gitter-netz (blau) zur Speicherung des Flow- Feldes.

Zur Berechnung und Darstellung des Flows wird die Strömungsfunktion benutzt, denn jedes zweidimensionale unkomprimierbare Feld lässt sich als Ring (Curl) einer skalaren Funktion – der Strömungsfunktion **S(x,y)** (häufig auch als **ψ** angeführt) multipliziert mit einem Vektor der 3. Dimension **z** – darstellen:

$$\mathbf{v}(x,y) = \nabla \times (S(x,y)\mathbf{z}) \qquad (3)$$

wobei die Curl- Funktion definiert ist als:

$$\nabla \times \mathbf{A} = \mathbf{a}_x \left(\frac{\partial A_z}{\partial y} - \frac{\partial A_y}{\partial z} \right) + \mathbf{a}_y \left(\frac{\partial A_x}{\partial z} - \frac{\partial A_z}{\partial x} \right) + \mathbf{a}_z \left(\frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y} \right).$$

Die Strömungsfunktion wurde 1781 von Lagrange entwickelt und findet in *Fluids Dynamics* häufig Anwendung. Sie ist definiert durch die Gleichungen über die Geschwindigkeiten **u** und **v** in x- bzw. y- Richtung:

$$u = \frac{\partial \Psi}{\partial y} \qquad v = -\frac{\partial \Psi}{\partial x} \qquad (4),$$

d.h. die Geschwindigkeit in x- bzw. y- Richtung an einem gegebenen Punkt berechnet sich aus den partiellen Ableitungen der Strömungsfunktion an diesem Punkt. Die Differenz

zwischen den Strömungsfunktions- Werten zweier Punkte ergibt den Flux über die Kante zwischen diesen Punkten. Außerdem ist die Kontinuität sichergestellt, da diese Definition der Strömungsfunktion die Kontinuitäts- Gleichung erfüllt:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

Deshalb ist sie für die Berechnung und Speicherung des Flows innerhalb der Flow- Tiles sehr gut geeignet.

Für die Anwendung der Strömungsfunktion bei Flow- Tiles wird der Curl- Operator aus Gleichung (3) diskretisiert, und an Stelle des Geschwindigkeitsfeldes werden die Werte der Strömungsfunktion an den Kanten des Rasters gespeichert. Die Geschwindigkeit (x',y') an einem Punkt (x,y) wird dann durch Ableitung der Strömungsfunktion berechnet:

$$\begin{aligned} \dot{x} &= S_{x,y-.5} - S_{x,y+.5} \\ \dot{y} &= S_{x+.5,y} - S_{x-.5,y} \end{aligned} \qquad (5)$$

Dabei wird ausgenutzt, dass die auf den Kanten des Rasters gespeicherten Strömungsfunktionswerte jeweils eine Einheit weit auseinander liegen (s. Abb.7). Die Ableitungsgleichungen (4) können dadurch umgeformt werden zu $(S_{x,y-0.5} - S_{x,y+0,5}) / 1$.

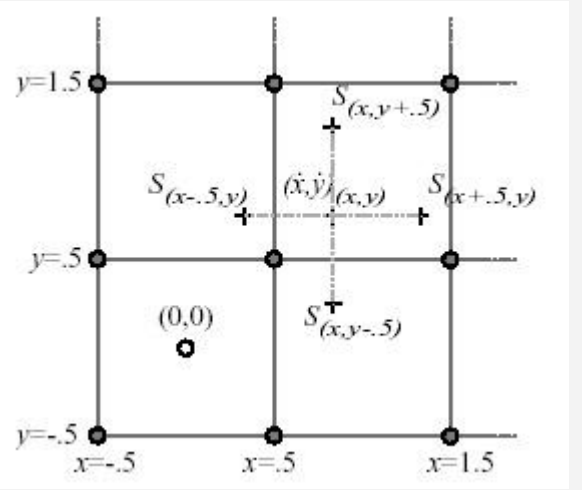


Abbildung 7: Strömungsfunktions- Methode zur Darstellung eines abweichungsfreien Geschwindigkeitsfeldes. Die Abb. zeigt die untere linke Ecke des Rasters.

Die Darstellung des Flows über die Strömungsfunktionswerte hat zwei Vorteile: niedrige Speicherkosten von $(n+1)^2$ für ein nxn Raster, und die vereinfachte Konstruktion von Tiles mit bestimmten Kanten- Fluxen. Denn der gesamte Flux über die Kante eines Tiles wird durch Addition (oder Subtraktion) der Strömungsfunktionswerte der Eckpunke bestimmt. All die Strömungsfunktionswerte, die dazwischen liegen, heben sich bei der Summation der Geschwindigkeiten über die Kante gegenseitig auf. Dadurch kann ein bestimmter Flux über eine Kante durch festlegen der Strömungsfunktionswerte der Ecken sichergestellt werden, ohne dass das Feld zwischen den Ecken betrachtet werden muss.

Die Strömungsfunktion wird für jedes Tile nur einmal angewendet, um es mit einem Flow- Feld zu füllen. Aus den gegebenen Eckgeschwindigkeiten und dem Flux des Tiles wird ein abweichungsfreies Geschwindigkeitsfeld berechnet und auf dem Raster des Tiles gespeichert. Das so erzeugte Geschwindigkeitsfeld ist wegen der Eigenschaften der Strömungsfunktion abweichungsfrei. Doch diese Methode funktioniert nur ab einer Tile-Größe von 3x3, d.h. ab einer Strömungsfunktions- Rastergröße von 4x4 (siehe Abb. 6). Alternativ dazu könnte man die Tiles auch von Hand oder mit einer anderen Flow-generierenden Methode (z.B. Kolmogorov Spektrum) füllen.

Ein Tile der Größe n_xxn_y benötigt ein Strömungsfunktions- Raster der Größe (-0.5,…, n_x+0,5) x (-0.5,…, n_x+0,5). Die Eckgeschwindigkeiten und Kanten- Fluxe legen die Werte der Strömungsfunktion an den Ecken des Tiles vollständig fest. So kann man mit Hilfe der für dieses Tile festgelegten Fluxe die Eckwerte berechnen:

$$\begin{aligned} S_{-.5,-.5} &= 0 \\ S_{n_x+.5,-.5} &= S_{-.5,-.5} + f_{bottom} \\ S_{-.5,n_y+.5} &= S_{-.5,-.5} - f_{left} \\ S_{n_x+.5,n_y+.5} &= S_{-.5,n_y+.5} + f_{top} \end{aligned}$$

und unter Verwendung der Geschwindigkeiten und Gleichung (5) können die anderen Werte um jede Ecke herum gesetzt werden. Beispielsweise:

$$S_{.5,-.5} = S_{-.5,-.5} + \dot{y}_{0,0}$$
$$S_{-.5,.5} = S_{-.5,-.5} - \dot{x}_{0,0}$$
$$S_{.5,.5} = S_{-.5,.5} + \dot{y}_{0,0}$$

wobei $x'_{0,0}$ und $y'_{0,0}$ die horizontale und vertikale Komponente der Geschwindigkeit an der unteren linken Tile- Ecke sind. Die Werte an den anderen Ecken werden ähnlich gesetzt.

Auf diese Weise erhält man die 16 Strömungsfunktions- Werte um die Ecken herum. Der Rest des Rasters wird durch ein *bi- kubisches Bezier- Patch* berechnet. Das Ergebnis ist ein glatter, kontinuierlicher Flow innerhalb des Tiles, wie beispielsweise das (3,3,1) Tile- Set auf Abbildung 5 zeigt.

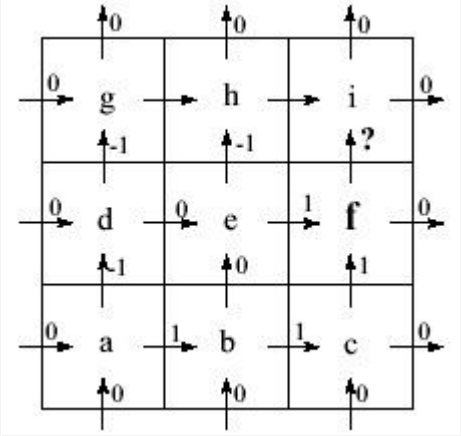
Das Paper von Chenney erläutert nur die Anwendung der Strömungsfunktions- Methode bei quadratischen Tiles, erwähnt aber auch, dass sie an nicht-quadratische Tiles angepasst werden könne.

3. Tiling

Als Tiling wird das Auslegen der Tiles zu einem kontinuierlichen Flow- Feld bezeichnet. Da das Ergebnis in größere Umgebungen integriert werden soll, darf es weder interne noch externe Grenzkonditionen verletzen. Dabei entsteht aus der Abweichungsfreiheitsbedingung kombiniert mit dem Gebrauch eines endlichen Tile- Sets das Problem ein Tiling auf erlaubte Weise erfolgreich zu beenden. Beispielhaft dafür ist das Tiling Abb.8, dessen Ziel es war ein 3x3 Feld mit einer Null- Grenzbedingung zu füllen. Und obwohl das bisher gelegte Tiling (a-e) abweichungsfrei ist, kann es nicht beendet werden, da das erforderliche Tile nicht im Set vorhanden ist. Folglich sind die Bedingungen, die durch die Tile- Kanten auf die Platzierung der Tiles erlegt wurden, nicht ausreichend, um die Fertigstellung eines gültigen Tilings mit einem festgelegten Tile- Set sicherzustellen.

Deshalb wurde zur Unterstützung des Anwenders der Flow- Tile- Benutzerschnittstelle der Tiling- Algorithmus entwickelt, welcher hilft Tiles auf einem rechteckigen Raster zu platzieren. Vorher können gewünschte Grenzkonditionen über den Flux der Tile- Kanten und die Eckgeschwindigkeiten auf dem Raster festgelegt werden, wobei diese im spezifizierten Tile- Set vorhanden sein müssen. Für den Fall, dass eine Anwendung nicht das komplette rechteckige Raster benötigt, sondern z.B. nur einen **+**- förmigen Teil davon, kann man durch das Null- Setzen der Fluxe und Eckgeschwindigkeiten der unerwünschten Region(en) erreichen, dass der Flow diese Grenzen nicht überquert.

Abbildung 8: Ein nicht zu beendendes Tiling. Es wurde das (3,3,1) Set benutzt, um die Tiles a-e zu platzieren. Nun müsste an Stelle f ein Tile eingesetzt werden, doch es gibt im Set kein Tile mit dem benötigten Flux von 2.



3.1 Die Selektionsfunktion

Das Herzstück des Tilings ist die Selektionsfunktion. Sie liefert zu einem gegebenen partiellen Tiling und einer ungefüllten Stelle im Raster eine Menge von Tiles, die an dieser Stelle passen, und stellt dabei sicher, dass das Tiling vervollständigt werden kann. Der Benutzer braucht dann nur noch eines dieser Tiles auszuwählen und einzufügen. Die Reihenfolge, in welcher die Stellen des Rasters gefüllt werden, spielt dabei keine Rolle.

Die Hintergrund- Idee des Selektions- Algorithmus ist, dass jedes platzierte Tile gerichtete Masse in seine Nachbarn drückt, welche sie an ihre Nachbarn weitergeben und so weiter über das ganze Raster verteilen. Auf diese Weise wird sichergestellt, dass auch die Stellen im Tiling, bei denen schon 3 Seiten mit Bedingungen belegt sind (wie in Abb. 8), mit einem Tile aus dem Set gefüllt werden können.

Zuerst wählt der Benutzer die Position, welche mit einem Tile gefüllt werden soll. Der Selektions- Algorithmus muss nun herausfinden, welche Flux- und Eckgeschwindigkeits-Werte hier möglich sind, um dann die entsprechenden, passenden Tiles zurückgeben zu können. Die möglichen Werte der Ecken ergeben sich aus den – vom Benutzer oder benachbarten Tiles – vorgegebenen Bedingungen. Die Eckgeschwindigkeiten des einzusetzenden Tiles müssen damit übereinstimmen, oder sind frei wählbar, wenn es dort keine Bedingungen gibt. Die möglichen Kanten- Fluxe werden jeweils von zwei Integer- Programmen berechnet. Abb.9 zeigt den Aufbau dieser Programme an einem Beispiel. Für jede freie Kante existiert eine Ungleichung, welche die Werte der Kante auf die im Set vorhandenen beschränkt. Und für jede unbelegte Tile- Position stellt eine Gleichung die Erfüllung der Abweichungsfreiheit sicher.

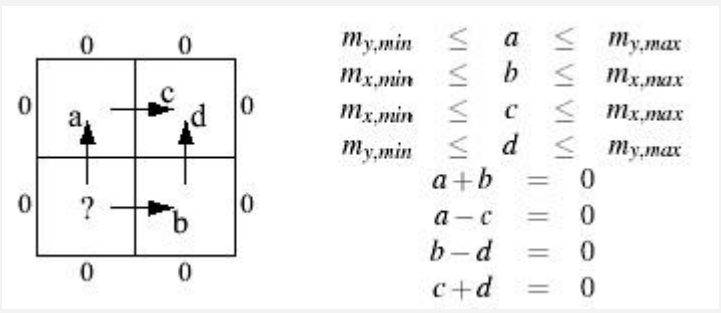


Abbildung 9: Beispiel eines Tiling- Rasters und den Bedingungen des Integer- Programms, welches passende Tiles für die linke untere Ecke liefert.
a, b, c und d sind die Flux- Werte über die Kanten. Die Ungleichungen unterstützen die Beschränkung auf das endliche Tile- Set, und die Gleichungen stellen die Abweichungsfreiheit jedes einzelnen Tiles sicher.

Die Bestimmung der möglichen Flux- Werte einer Kante erfolgt in zwei Schritten. Beim ersten Schritt maximiert das eine Integer- Programm den Flux, während das zweite ihn minimiert. Danach sind die Grenzwerte bekannt, und aus dem Flow- Tile- Set werden alle Tiles herausgesucht, deren Flux innerhalb dieser Grenzen liegt. Doch manche dieser Tiles sind trotzdem nicht akzeptabel. Z.B. zeigt Abbildung 10 eine Stelle, für welche die passenden Tiles aus einem Set, dessen Flux-Werte von -2 bis 2 reichen, herausgesucht werden sollen. Der erste Schritt zur Minimierung und Maximierung der Fluxe a und b ergibt a = [0,2] und b = [0,2], da ein Flux von beispielsweise a = -1 nur möglich wäre, wenn für b ein Wert von 3 im Tile- Set vorhanden wäre. Nun ist aber ein Wert von b = 2 an der oberen Kante nicht zulässig, wenn die rechte Kante einen Flux von a = 2 aufweist. Dies wird im 2. Schritt sichergestellt: Jedes Tile, das beim ersten Schritt zurückgegeben wurde, wird nun auf seine Machbarkeit getestet.

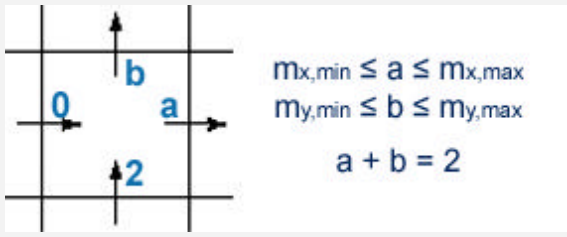


Abbildung 10: An diesem Platz sind die Flux- Werte der unteren und linken Kante fest, die möglichen Werte (a und b) für die rechte und obere müssen mithilfe des Programms bestimmt werden.
Als Ergebnis liefert es: a=2, b=0 oder a=1, b=1 oder a=0, b=2.

Zu dem Programm werden zusätzliche Bedingungen hinzugefügt, die das jeweilige Tile repräsentieren, und dann überprüft, ob eine gültige Lösung existiert. Falls ja, wird es als eines der möglichen Tiles für diese Position zurückgegeben. Beispielsweise würden für das Tile mit (m_{bottom}, m_{right}, m_{top}, m_{left}) = (2,2,2,0) die Bedingungen a = m_{right} = 2 und b = m_{top} = 2 zum Programm Abb.10 hinzugefügt, und dann getestet, ob das Gleichungssystem eine Lösung enthält (was es natürlich nicht tut). Nach dem Testen aller Tiles bestände das Ergebnis für dieses Beispiel aus den Tiles (2,2,0,0), (2,1,1,0) und (2,0,2,0).

Der wichtigste Bestandteil des Selektions- Algorithmus ist die Lösung der Integer-Programme, wofür eine Standart Simplex- Methode *linear program solver* benutzt wird. Integer- Programme sind hier besonders geeignet, da die den Tiles auferlegten Bedingungen als Gleichungen über die (beschränkten) Werte der Kanten darstellbar sind (s. Abb. 9). Außerdem gehören sie zu den *network flow problems* [Van01], die dank dem Integral-Theorem nur Integer- Werte zurückgeben, und sich deshalb mit dem *Osi/Clp Linear Program Solver* des *COIN- Package* [LH03] lösen lassen, der zu den weniger teuren linearen Programm- Lösern gehört und stattdessen sogar Hot- Start- Lösungen unterstütz. Das bedeutet, dass Aufgaben, zu denen eine Lösung existierte, die aber nun wegen Änderungen an der Zielfunktion nicht mehr gültig ist, mit bedeutend niedrigeren Kosten wieder gelöst werden können, als wenn man sie von Anfang neu an lösen müsste. Und genau diese Situation tritt im Selektions- Algorithmus jedes Mal auf, wenn für eine bestimmte Tile- Position die möglichen, dort passenden Tiles ausgewählt werden sollen: es wird nur die Zielfunktion geändert, nicht aber die Bedingungen, denn diese hängen von den Eigenschaften des Tilings ab und nicht von einer bestimmten Kante.

3.2 Random- Tiling

Neben dem interaktiven Tiling gibt es auch eine automatisierte Methode Geschwindigkeitsfelder zu erzeugen: die *Random- Tilings*. Zur Erstellung eines solchen Tilings wird in einer scan-line Reihenfolge jeweils eine Position des Tiling- Raster ausgewählt, die Menge der Tiles, die die dortigen Bedingungen erfüllen, berechnet und dann per Zufall eines daraus ausgewählt. Der Gebrauch von Random- Tilings ist bei natürlichen Effekten wie Windfeldern interessant, oder auch für Benutzer, die sehen möchten welche Geschwindigkeitsfelder aus ihren Vorgaben entstehen können.

3.3 Tiling- Kosten

Die Kosten des Tilings genau zu bestimmen ist schwierig. Die Anzahl der Variablen jeden Programms ist bekannt (bei einem nxn Tiling sind es 2n(n+1), teilweise mit festen Werten), aber die Anzahl der Constaints hängt ab von den freien Tile- Plätzen im Tiling, und der Gebrauch des *simplex solvers* macht es schwer Performance- Vorraussagen zu machen. Zusätzlich dazu hängt die Anzahl der Programme, die im 2. Schritt gelöst werden müssen, davon ab, wie viele Tiles im ersten Schritt ausgewählt wurden. Ein 10x10 Random- Tiling-Algorithmus benötigt die Lösung von über 650 Programmen, die in weniger als 1.5 Sekunden berechnet werden. Ein 4x4 Random- Tiling benötigt 0.077 Sekunden (auf einem 1.3GHz PC), und ein 32x32 Tiling braucht 113 Sekunden.

4. Einfügen in Umgebungen

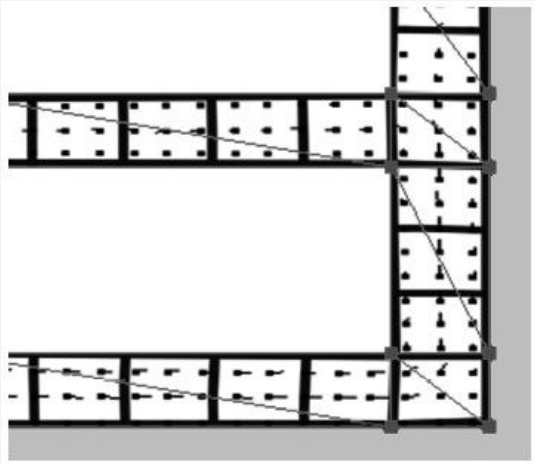


Abbildung 11 zeigt einen Teil des Stadt- Modells, das innerhalb eines polygonalen Modells (dunkelgrau) liegt und das Tiling (schwarze Linien). Die Punkte des Models wurden mithilfe der Flow- Koordinaten mit Punkten des Tilings identifiziert. Diese Flow- Koordinaten erlauben es auch die Geschwindigkeiten vom Tiling in die Umgebung zu übertragen.

Sobald das Tiling fertig ist, kann es in eine Umgebung – etwa ein Spiel oder eine interaktive Schnittstelle – eingefügt werden (s. Abb. 11). Dies geschieht, indem Punkte in der Umgebung mit Punkten des Tilings identifiziert werden, ähnlich wie bei *texture mapping*. Die Gebiete, auf denen durch Grenzkonditionen kein Flow erzeugt wurde, (‚weiße Stellen') werden zumeist mit Hindernissen der Umgebung identifiziert, wie z.B. eine Häuserwand. Um ein gutes Ergebnis zu erhalten sollte das Tiling sehr passgenau eingefügt werden, damit zwischen ihm und der Umgebung keine Spalten entstehen. Risse entlang von Tile- Grenzen sind jedoch erlaubt – Abb.1 zeigt einen solchen an der Stelle, an der der Fluss sich teilt –, um zu verhindern, dass Tiles unsichtbar unter der Umgebungsoberfläche platziert werden, und um scharfe Ecken der Umgebung zu umkleiden, ohne das Raster zu sehr zu verzerren.

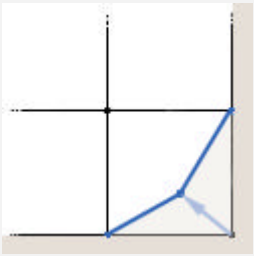


Abbildung 12: Mögliche Verzerrung des Rasters (blau) beim Einfügen in eine Umgebung (braun). Die Masse in dem verzerrten Raum rechts unten wird komprimiert.

Zur Identifikation der Umgebungs- Punkte mit denen des Tilings werden Flow- Koordinaten benutzt. Diese können so definiert werden, dass sie das Raster verformen (s. Abb.12, 13). Für die Masse, die durch das Flow- Feld dargestellt wird, heißt das, dass sie komprimiert oder dekomprimiert wird. Dies ist für komprimierbare Massen – wie einer Menschenmenge, die einfach näher zusammenrückt – möglich, aber nicht bei unkomprimierbaren Massen. Bei Wasser beispielsweise müsste in einem komprimierten Feld (wie bei Abb.12 rechts unten) ein Teil der Flüssigkeit entweder unter die 2D- Oberfläche gedrückt werden, oder als Welle über sie.

Durch eine Verformung des gesamten Rasters – wie beispielhaft in Abb.13 gezeigt – kann trotz der 2-dimensionalität des Rasters ein 3-dimensional wirkender Flow erzeugt werden. Dieser Effekt ist bei Abb.1 sehr gut zu erkennen, denn der Fluss wirkt keinesfalls 2D, obwohl er durch das links daneben abgebildete 2D- Raster erzeugt wird. Das kommt daher, dass das Raster in der 3D-Umgebung auf eine flache Oberfläche projiziert wird – bei dem Fluss-Beispiel auf die Wasseroberfläche, und bei dem Beispiel der Menschenmenge in einer Stadt auf die Straßen (s. Abb.15) – auf welcher dann das Geschwindigkeitsfeld die Effekte – Blätter bzw. Menschen – steuert.

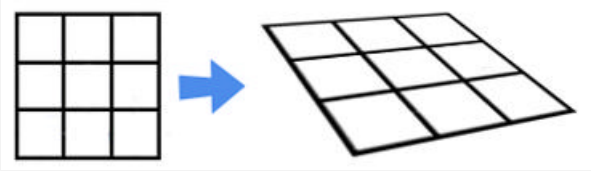


Abbildung 13: Durch Transformation des Rasters während des Einfügens in eine Umgebung kann der Eindruck von 3-Dimensionalität auch trotz der 2D- Darstellung des Flow-Feldes erzeugt werden.

Das Einfügen eines Tilings in eine Umgebung (Welt) erfolgt folgendermaßen: Der Raum der Welt wird in Dreiecke unterteilt, wobei jedes einzelne jeweils einem Dreieck des Tilings entspricht. Um die Geschwindigkeit an einem Punkt (x,y) der Welt zu bestimmen, wird dieser zuerst als Punkt (α,β,γ) des Baryzentrischen Koordinatensystems ausgedrückt, welches durch die Dreiecksaufteilung der Welt definiert ist. Dieser Punkt wird dann mittels Flow- Koordinaten in einen Punkt (u,v) des Tilings übersetzt. Dies ist grundsätzlich genau wie bei *texture mapping*, nur dass nun anstelle einer Farbe die Geschwindigkeit zurück in die Welt übertragen werden muss:

Die Geschwindigkeit (u',v') des Flows wird zunächst in das durch die Flow-Koordinaten (α',β',γ') definierte Baryzentrische Koordinatensystem umgerechnet:

$$\dot{\alpha} = \frac{\partial \alpha}{\partial u}\dot{u} + \frac{\partial \alpha}{\partial v}\dot{v}$$, äquivalent für β' und γ'.

Die Terme **δα/ δu** usw. sind partielle Ableitungen des Baryzentrischen Koordinatensystems in Hinsicht auf das Flow- Koordinatensystem. Dafür werden α, β und γ als Gleichungen über u, v und den Flow- Koordinaten dargestellt und dann partiell abgeleitet. Um ein schnelles Such-verfahren zu ermöglichen, können die Ergebnisse direkt mit dem Dreieck gespeichert werden, und dann letztendlich in die Geschwindigkeit in der Welt übersetzt werden:

$$\dot{x} = \frac{\partial x}{\partial \alpha}\dot{\alpha} + \frac{\partial x}{\partial \beta}\dot{\beta} + \frac{\partial x}{\partial \gamma}\dot{\gamma}$$, ähnlich für y'.

δx/ δα wird berechnet, indem die Koordinaten der Welt (x,y) zu Ausdrücken über α, β und γ umgeformt und dann partiell abgeleitet werden. Das Ergebnis ist dann Teil der Welt-Koordinaten.

Der letzte Teil der Spielentwicklung ist eine *runtime engine*, die es versteht mit Flow-Koordinaten und eingefügten Tilings zu operieren und im Paper nicht weiter erläutert wurde.

5. Die Benutzerschnittstelle

Flow- Tiles eignen sich dafür große Geschwindigkeitsfelder zu entwerfen, die spezifische Grenzkonditionen erfüllen. Um die Anwendung zu vereinfachen, wurde eine WYSIWYG- Benutzerschnittstelle und ein interaktiver Tiling- Editor entworfen, der in ein Toolset für Spielentwicklung integriert ist. Diese Editor (s. Abb.14) ermöglicht es ein Tile-Set zu laden, die Region zu bestimmen, welche mit Tiles versehen werden soll, und die Grenzkonditionen zu setzen. Dann kann der Benutzer das Tiling entwerfen, indem er zuerst auf eine leere Tile- Position klickt und dann eines der Tiles wählt, die das System anbietet. Da der Selektionsprozess aus Abschnitt 3 angewendet wird ist sichergestellt, dass das Tiling beendet werden kann. Sobald das Tiling fertig ist, kann es in ein anderes Tool exportiert werden, das es in eine Umgebung einbindet (s. Abb. 11, Abschnitt 4).

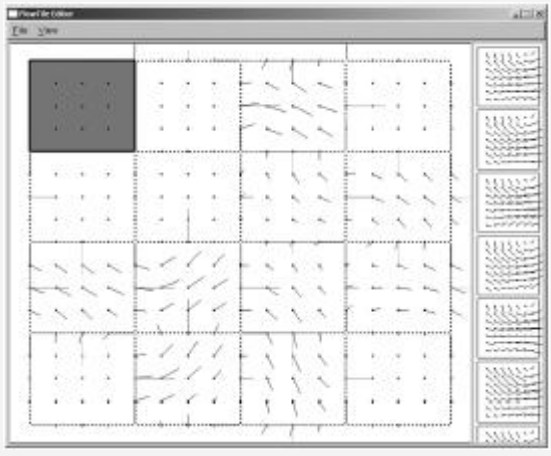


Abbildung 14: Ein 4x4 Tiling während des Aufbaus in einem Editor. Der Benutzer setzt Bedingungen auf Flux und Eckgeschwindigkeiten der Tiles, und das Programm berechnet das Tile-Set und füllt die einzelnen Tiles. Wenn eine leere Tile- Position gewählt wurde (hier die linke obere Ecke), zeigt der Editor auf der rechten Seite eine Reihe von Tiles an, die an diese Stelle passen, und der Benutzer braucht nur eines davon auszuwählen. Die hier gewählte Position hat schon festgesetzte Werte für die linke und die obere Kante, und für drei der Eckgeschwindig-keiten, deshalb sehen sich die angebotenen Tiles sehr ähnlich.

6. Beispiele

Chenney stellt in seinem Paper zwei Beispiele für Standart- Flow- Tile- Anwendungen dar – das Model einer Stadt, bei der Flow- Tiles die Bewegung der Menschen durch die Straßen bestimmen, und einen Fluss, der in seinem Flussbett um einen Brückenpfeiler herum fließt und sich dann teilt –, und ein Beispiel für zeit- variierende Flows – wirbelnder Nebel –.

Abbildung 15: Menschen bewegen sich von Flow- Tiles geführt durch die Stadt. Interne Grenzkonditionen halten sie davon ab durch Wände zu laufen.

Die Straßen der Stadt (siehe Abb.15) wurden mit einem (5,5,5) Tile- Set ‚gepflastert', d.h. es gibt jeweils fünf mögliche Kanten- Fluxe und fünf Eckgeschwindigkeiten. Diese Tiles führen eine Menschenmenge durch die Straßen, wobei ihre Bewegungsrichtung durch die in den Tiles gespeicherte Geschwindigkeit bestimmt wird. Aufgrund der Beschaffenheit der Flow- Tiles benötigen die Agenten keine Kollisionskontrolle, denn unter der Vorraussetzung, dass sie ohne Überschneidungen und mit etwas Sicherheitsabstand zueinander starten und dass das Feld fast unkomprimiert ist, wird der Flow sie von anderen Agenten und den Wänden der Gebäude fern halten. Jedoch können die Agenten (wegen der Beschaffenheit der Tiles) nicht miteinander kommunizieren, was auch für zahlreiche Anwendungen, z.B. die Animation einer Notfall- Evakuierung oder einer panischen Menschenmenge, nicht notwendig ist. Für den Fall, dass dennoch Interaktion erfordert wird, können lokale Interaktions- Gebiete eingefügt werden, in welchen die Agenten nicht mehr dem Flow folgen, sondern nach einer anderen Methode miteinander kommunizieren. Außerdem ist eine Folge der Flow- Tiles, dass sich die Stromlinien – und somit die Wege zweier Agenten – niemals kreuzen (siehe z.B. Abb.5 Tile 5 und 6). Dies könnte durch zwei – auf einem Platz – übereinander gelegte Tiles erreicht werden, doch für diesen Bereich wäre dann eine Kollisionskontrolle notwendig.

Das Fluss- Beispiel (Abb.1) führt die Blätter auf der Flussoberfläche um einen Brückenpfeiler herum. Hier wurde ein Tile- Set mit fünf Flux- Optionen in Flussrichtung (mit/ gegen den Strom), drei Flux- Optionen in seitlicher Richtung und fünf möglichen Eckgeschwindigkeiten verwendet. Um die Verwirbelung hinter dem Brückenpfeiler zu bewirken, wurden dort entsprechende Bedingungen für den Flux und die Eckgeschwindigkeiten gesetzt. Die Gabelung des Flusses wurde erzeugt, indem das Tiling entlang einer internen Kante durchgetrennt und die zwei Hälften dann auseinander gezogen wurden. Die Simulation eines relativ großen Flusses wie diesem wäre für eine Echtzeit- Ausführung zu teuer, besonders bei den anderen Anforderungen an Prozessor- Zeit und -Speicher, die bei einem Computerspiel oder einer interaktiven Anwendung hinzukommen. Flow- Tiles hingegen sind effizient zu speichern (brauchen nur 20% des Platzes, der ohne zum Speichern des Feldes ohne mehrmals benutzte Tiles benötigt würde) und die Geschwindigkeit der Objekte ist schnell zu berechnen.

Flow- Tiles können erweitert werden, um zeit- variierende Flows zu kreieren. Um einen solchen Flow zu erzeugen, werden mehrere unterschiedliche Tilings des gleichen Gebietes erstellt, welche dann hintereinander geschaltet werden. Für den fließenden Übergang von einem Tiling zum nächsten wird zwischen den Tilings interpoliert. Dies funktioniert, da die Linearkombination zweier abweichungsfreier Flows wieder abweichungsfrei ist. Der

Benutzer kann die Tilings selbst erstellen und eine Reihenfolge angeben, in welcher sie dann hintereinander geschaltet werden. Für längere Animationen wird dann eine Schleife eingerichtet. Alternativ dazu können Random- Tilings erzeugt und in zufälliger Reihenfolge angewendet werden, um den zeit- variierenden Flow darzustellen. Ein solcher Flow wurde auch zur Animation des wirbelnden Nebels (s. Abb. 16) verwendet.

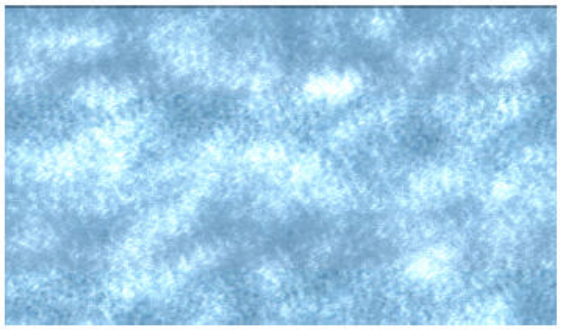


Abbildung 16: Die Animation wirbelnden Nebels generiert durch zeit- variierende Flow- Tilings.

7. Fazit

Flow- Tiles ist eine Methode zur Generierung eines 2dimensionalen Geschwindigkeits-feldes, das besonders für große Flows, die bestimmte Eigenschaften erfüllen und eine Fläche (z.B. Straßen oder eine Wasseroberfläche) einer interaktiven Umgebungen beleben sollen, geeignet ist. Der erste Schritt ist die Erzeugung eines Tile- Sets mit ausschließlich abweichungsfreien Tiles, für das der Benutzer die Unter- und Oberwerte der Kanten- Fluxe und die möglichen Eckgeschwindigkeiten festlegt. Danach wird die Tiling- Fläche definiert, die mit Tiles gefüllt werden soll. Der Benutzer wird beim Tiling von der Benutzerschnittstelle unterstützt, die mithilfe der Selektionsfunktion die jeweils zu einer Tiling- Position passenden Tiles aus dem Tile- Set bestimmt und dabei sicherstellt, dass das Tiling mit den vorgegebenen Tiles vervollständigt werden kann. Sobald das Tiling beendet ist, kann das Flow- Feld in eine Anwendung integriert werden, wobei darauf geachtet werden muss, dass zwischen Tiling und Anwendung keine Spalten entstehen und dass das Tiling nicht zu sehr verzerrt wird.

Aus der Art und Weise, mit der aus Flow- Tiles ein Geschwindigkeitsfeld erzeugt wird, entstehen sowohl Vor- als auch Nachteile:

Wegen der quadratischen Form der Tiles können für das Tiling nur rechteckige Raster benutzt werden, was – trotz der Möglichkeit Teile des Rasters mit einem Null- Flow zu belegen – bedeutet, dass es von der Form einer Fläche abhängt, ob sie ohne zu große Verzerrung mit Tiles bekleidet werden kann. Wegen der gleichmäßigen Struktur des Rasters können bei kleineren Tile- Sets mit wenigen Eckgeschwindigkeits- Optionen leicht Artefakte erkannt werden. Bei vielen Eck- Optionen wächst das Tile- Set allerdings sehr schnell, und wenn es zu groß wird entstehen beim Selektions- Schritt des interaktiven Tilings Probleme, weil es für eine gegebene Tile- Position sehr viele potentielle Tiles geben kann. Die Strömungsfunktion, die zum Füllen der Tiles benutzt wird, hat den Nachteil, dass damit keine kleineren Turbolenzen innerhalb eines Tiles erzeugt werden können. Doch man kann auch andere Methoden benutzen, um die Tiles zu füllen; die Definition des Tile- Sets, der Eckgeschwindigkeiten, Fluxe und Abweichungsfreiheit funktioniert trotzdem genauso. Als Nachteil der Flow- Tiles könnte man auch sehen, dass der Flow – anders als z.B. bei Simulationsmethoden – ohne physikalischen Hintergrund gestaltet wird, und durch Grenzkonditionen und Abweichungsfreiheit nur die grundlegenden Bedingungen erfüllt.

Doch genau dies stellt auch die Stärke der Flow- Tiles dar: der Flow kann so entworfen werden, wie man es wünscht. Das Ergebnis ist planbar, anders als bei Simulation. Außerdem kann durch die Wiederverwendung der einzelnen Tiles ein großer Flow gestaltet, effizient gespeichert und zur interaktiven Nutzung in Echtzeit berechnet werden, was bei anderen Methoden nicht möglich ist. Die Grenzbedingungen stellen sicher, dass der Flow in eine Anwendung integriert werden kann, und die Abweichungsfreiheit ermöglicht es eine Vielzahl

von natürlichen Effekten darzustellen. Zusätzlich wird der Benutzer bei der Gestaltung des Flow- Feldes von der WYSIWYG- Benutzerschnittstelle unterstützt. Und der Tiling Selektions- Algorithmus funktioniert auch bei nicht-quadratischen Tiles, wodurch aus jeder Art von polygonalen Tiles – einschließlich Dreiecken – abweichungsfreie Flows erstellt werden können. Dadurch werden Artefakte verringert, und die Tiling- Fläche muss nicht mehr quadratisch sein. Doch die Herausforderung dabei ist die Anpassung der Technik zum Füllen der Tiles und zum Überwachen der Abweichungsfreiheit.

Chenney konzentriert sich in seinem Paper sehr auf den Entwurf der Flow- Tiles und des Tilings, und gibt die dafür notwendigen Berechnungszeiten an, doch die der *runtime engine*, mit Flow- Koordinaten und eingefügten Tilings operiert und so die einzelnen Objekte innerhalb des Flows steuert – wie die Blätter auf dem Fluss und die Menschen in der Stadt –, werden im Paper nicht weiter erläutert. Er erwähnt nicht, was die Positions-/ Bewegungsberechnung eines Objektes kostet, und ob ab einer bestimmten Anzahl von animierten Objekten Probleme gibt die Berechnung in Echtzeit auszuführen.

Außerdem ist das Paper in Bezug auf den zweiten Schritt des Selektionsalgorithmus nicht ganz eindeutig/klar. Denn wenn beim Erstellen des Tile- Sets darauf geachtet wurde, dass nur abweichungsfreie Tiles darin vorhanden sind, müsste eigentlich schon der erste Schritt des Selektionsalgorithmus das gewünschte Ergebnis liefern. In diesem Schritt werden mithilfe der Ungleichungen, die die Werte der Kanten- Fluxe auf die im Set vorhandenen beschränken, und der Gleichungen, die die Erfüllung der Abweichungsfreiheit sicherstellen (wofür die an der jeweiligen Tile- Stelle vorgegebenen Flux- Werte in die Gleichungen eingesetzt wurden), die Intervalle [min, max] der möglichen Werte für die noch offenen Fluxe berechnet. Daraufhin werden alle Tiles mit entsprechenden Flux- Werten aus dem Tile- Set herausgesucht, d.h. bei dem Beispiel Abb.10 werden aus dem Set alle Tiles mit $m_{left} = 0$, $m_{bottom} = 2$, $m_{right} = [0,2]$, $m_{top} = [0,2]$ herausgesucht. Im Paper wird der zweite Selektionsschritt damit begründet, dass nicht zugleich $m_{right} = 2$ und $m_{top} = 2$ sein dürfe, doch in einem Tile- Set mit ausschließlich abweichungsfreien Tiles wäre ein solches (2,2,2,0) Tile auch gar nicht vorhanden und der zweite Selektionsschritt somit unnötig.

Abschließend möchte ich betonen, dass diese Technik nicht für jede Anwendung optimal ist. Sie eignet sich besonders für Flow- Animationen auf ebenen Flächen, die bestimmte Bedingungen erfüllen sollen, und weniger für physikalisch korrekte 3D- Simulationen. Wenn man sich zur Lösung eines Problems zwischen Flow- Tiles und anderen Techniken entscheiden will, sollte man sich der Vor- und Nachteile (s. o.) bewusst sein, und abwägen, ob Flow- Tiles den gewünschten Effekten entspricht.

8. Quellenangabe

Hauptquelle:

- Flow Tiles, SCA 2004. Stephen Chenney, University of Wisconsin. http://www.cs.wisc.edu/~schenney/research/replication/flowtiles/index.html

Zusätzliche Quellen:

- Wikipedia. Online im Internet. URL: http://wiki.4hv.org/index.php/Curl_Operator_and_Stokes_Theorem http://en.wikipedia.org/wiki/Stream_function
- Scienceworld. Online im Internet. URL: http://scienceworld.wolfram.com/physics/StreamFunction.html
- http://www.ae.su.oz.au/aero/fprops/poten/node10.html
- http://astron.berkeley.edu/~jrg/ay202/node96.html