Рис. 5. Тороидальное копирование

Изменим на графическом процессоре текстурные координаты триангулированной сетки по формулам: $\mathbf{u=u+}l_x$, $\mathbf{v=v+}l_y$. После чего текстура на сетке отобразится со сдвигом (рис. 6).

Соответственно, при сдвиге еще на один шаг в этом же направлении новые столбец и строка скопируются на позицию **{1, 1}** и циклически сдвинутся на 2 позиции, когда позиция $\mathbf{x_c=w_t}$, по формуле (3) она обнулится, то же произойдет и с $\mathbf{y_c=w_t}$, то есть это копирование циклически будет повторяться бесконечно.

В данной статье были рассмотрены основные принципы тороидального копирования в алгоритмах моделирования рельефа местности с помощью вложенных регулярных сеток. В приведенных примерах были матрицы малых размеров, но на практике используют матрицы с **n=255** и выше. Отсюда можно рассчитать, что при копировании

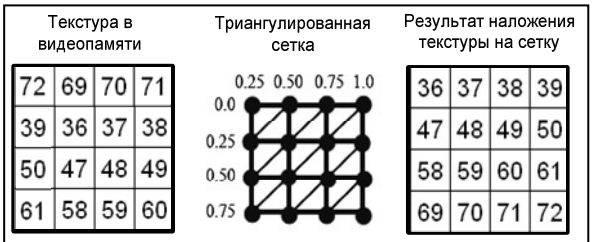


Рис. 6. Наложение текстуры после тороидального копирования

матрицы высот в текстуру полностью будет перемещено $\mathbf{n^2=65025}$ элементов, а с помощью тороидального копирования $\mathbf{n\cdot 2=65025}$ элементов, что в **127,5** раза меньше.

Литература

1. Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids Frank Losasso Hugues Hoppe. URL: http://research.microsoft.com/en-us/um/people/hoppe/geomclipmap.pdf (дата обращения: 19.07.2009).

2. GPU based clipmaps Implementation of Geometry Clipmaps for terrain with non-planar basis Anton Fruhstuck. URL: http://www.cg.tuwien.ac.at/research/publications/2008/fruehstueck-2008-gpu/fruehstueck-2008-gpu-paper.pdf (дата обращения: 19.07.2009).

3. Arul Asirvatham and Hugues Hoppe. Terrain rendering using GPU-based geometry clipmaps. In M. Pharr and R. Fernando, editors. URL: http://research.microsoft.com/en-us/um/people/hoppe/gpugcm.pdf (дата обращения: 19.07.2009).

4. Interactive Terrain Rendering: Towards Realism with Procedural Models and Graphics Hardware Dipl.–Inf. Carsten Dachsbacher Erlangen–2006. URL: http://www.vis.uni-stuttgart.de/~dachsbcn/publications.html (дата обращения: 19.07.2009).

5. URL: http://www.nvidia.com/page/technologies.html (дата обращения: 19.07.2009).

МОДЕЛЬ КОНСТРУКТИВНО-УНИВЕРСАЛЬНОГО АВТОМАТА

В.В. Дрождин, к.т.н.; М.В. Жуков
(Пензенский государственный педагогический университет им. В.Г. Белинского, drozhdin@spu-penza.ru)

Рассмотрена модель конструктивно-универсального автомата, определены базовые элементы, из которых конструктивно-универсальный автомат может сконструировать любой другой автомат, предложен способ спецификации процедуры сборки автомата.

Ключевые слова: *клеточный автомат, универсальный автомат, конструктивно-универсальный автомат, описание автомата, конструирование автомата.*

В своей работе по теории создания самовоспроизводящихся автоматов фон Нейман задался вопросом конструктивной универсальности, то есть способности автомата, заданного надлежащим образом и имеющего все необходимое сырье для построения других автоматов, сконструировать любой другой автомат [Дж. фон Нейман]. Более того, фон Нейману в рамках клеточной модели удалось построить логическую модель такого автомата. Однако она носит весьма специфический характер вследствие выбора первичных элементов, поэтому оказалась малопригодной для решения практических задач.

Обозначим через **A** множество всевозможных автоматов. Каждый автомат строится из элементов четырех типов: рецепторов, тела автомата, соединительных элементов и эффекторов.

Рецепторы (входы) – элементы автомата, которые способны воспринимать сигналы из внешней среды или от эффекторов.

Множество рецепторов обозначим через **I={i}**.

Эффекторы (выходы) – элементы автомата, передающие сигнал, сгенерированный телом автомата, во внешнюю среду или рецепторам. Выходной сигнал является реакцией автомата на внешнее воздействие или на изменение своего внутреннего состояния. Множество эффекторов обозначим через **O={o}**.

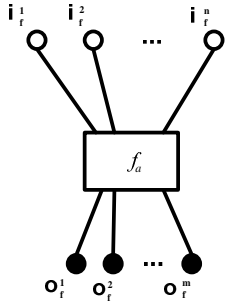
Соединительные элементы – элементы автомата, реализующие каналы связи между парами элементов. Среди соединительных элементов выделим элементы двух типов – простые соединения и адаптеры. ***Простое соединение*** может соединять рецептор с телом автомата или тело автомата с эффектором и передает сигнал от входа к выходу без каких-либо изменений. ***Адаптеры*** – более сложные соединители, но их описание зададим при конструировании *конструктивно-универсального автомата* (КУА). Множество соединительных элементов обозначим через **C={c}**.

Тело автомата задается в виде $\mathbf{f}_m^n : \mathbf{X} \rightarrow \mathbf{Y}$, где $\mathbf{X}=[\mathbf{i}_f^1, \mathbf{i}_f^2, ..., \mathbf{i}_f^n]$, $\mathbf{i}_f^j \in \mathbf{I}$, $\mathbf{j}=\overline{\mathbf{1,n}}$ – вектор входных воздействий, поступающих от рецепторов; $\mathbf{Y}=[\mathbf{o}_f^1, \mathbf{o}_f^2, ..., \mathbf{o}_f^m]$, $\mathbf{o}_f^j \in \mathbf{O}$, $\mathbf{j}=\overline{\mathbf{1,m}}$ – вектор выходных воздействий автомата, воспроизводимых эффекторами. Отношение соответствия между рецептором и положением во входном векторе, а также между положением в выходном векторе и эффектором задается простым соединением. Тело автомата переходит в возбужденное состояние (генерирует реакцию), если каждый его рецептор находится в возбужденном состоянии (отметим, что среди всех сигналов будем выделять специальный ***null-сигнал***, указывающий на отсутствие информации; рецепторы, способные фиксировать такой сигнал и переходить под его воздействием в возбужденное состояние, назовем ***сверхчувствительными рецепторами*** и обозначим как $\overline{\mathbf{i}}$, $\overline{\mathbf{i}} \in \mathbf{I}$).

Будем считать, что конструктивные элементы всегда имеются в требуемом количестве.

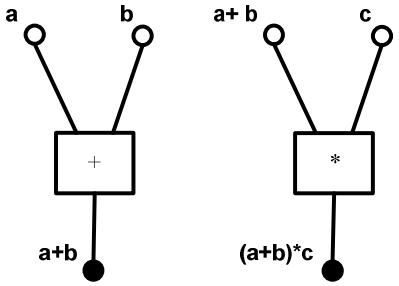
Каждому автомату сопоставим уникальный идентификатор $\mathbf{id} \in \mathbf{Id}$, где **Id** – множество идентификаторов. Тогда множество возможных автоматов задается пятеркой $\mathbf{A}=\{\mathbf{a} \mid \mathbf{a}=<\mathbf{I_a}, \mathbf{O_a}, \mathbf{f_a}, \mathbf{C_a}, \mathbf{id_a}>, \mathbf{C_a} \subseteq \mathbf{C}\}$.

Представим автомат в виде схемы:



Примечание: здесь и далее в схемах обозначения структурных элементов автомата зададим в виде: ○ – рецептор; ● – эффектор; ▭ – тело автомата; ———– простой соединитель.

В множестве **A** выделим подмножество $\mathbf{A_0}$ ***элементарных автоматов***, то есть минимальных автоматов, из которых строятся сложные автоматы. Каждый автомат $\mathbf{A_0}$ состоит только из рецепторов, тела автомата, эффекторов и простых соединений. Например, элементарные автоматы для вычисления выражения **[(a+b) * c]** представляются в виде



Конструирование КУА начнем с создания подсистемы – ***реестра автоматов*** **R**, представляющего собой бесконечно расширяемый словарь автоматов, то есть $\mathbf{R} \subset \mathbf{A}$. На множестве **R** определим две операции:

– ***get***, возвращающую из **R** автомат с $\mathbf{id}=\mathbf{id_a}$, задаваемую в виде $\mathbf{get: id_a \rightarrow a}$;

– ***contains***, проверяющую, содержится ли автомат с идентификатором $\mathbf{id_a}$ в **R**, задаваемую в виде $\mathbf{contains: id_a \rightarrow true| false}$.

В начальном состоянии реестр **R** будет состоять из множества первичных элементов. По мере функционирования КУА в **R** будут включаться все сконструированные автоматы.

Функционирование КУА представим отображением **Cu** вида $\mathbf{Cu: d(a_c) \rightarrow a_c}$, где $\mathbf{a_c}$ – конструируемый автомат; $\mathbf{d(a_c)}$ – описание автомата $\mathbf{a_c}$.

Для определения $\mathbf{d(a_c)}$ введем понятие обособленного элемента.

Обособленный элемент **a'** – это композиция элементов из **R**, обладающая свойствами *целостности* (рассматривается и используется как элемент) и *локальности* (реализует законченную последовательность действий).

Следовательно, обособленный элемент **a'** является автоматом.

Каждый обособленный элемент **a'** – элемент множества **A** ($\mathbf{a'} \in \mathbf{A}$), но, в отличие от $\mathbf{a_c}$, элемент $\mathbf{a'} \notin \mathbf{R}$. ***Описание обособленного элемента*** **d(a')** задается четверкой $\mathbf{d(a')}=<\mathbf{Id_{a'}}, \mathbf{Rel_{a'}}, \mathbf{I_{a'}}, \mathbf{O_{a'}}>$, где $\mathbf{Id_{a'}}=\{\mathbf{id_{a'}}\}$ – множество идентификаторов элементов, из которых строится **a'**; $\mathbf{I_{a'}}$ – рецепторы автомата **a'**; $\mathbf{O_{a'}}$ – эффекторы автомата **a'**; $\mathbf{Rel_{a'}}$ – матрица смежности ориентированного графа, вершинами которого являются множество рецепторов и эффекторов **a'**, а также множество рецепторов,

эффекторов и идентификаторов всех элементов из **a'**.

Описание автомата $\mathbf{a_c}$ задается пятеркой $\mathbf{D(a_c)=<h_{a_c}, D_{a_c}, \theta_{a_c}, I_{a_c}, O_{a_c}>}$, где $\mathbf{D_{a_c}=\{d(a')\}}$ – множество описаний обособленных элементов автомата $\mathbf{a_c}$; $\mathbf{h_{a_c}}$ – шаблон $\mathbf{a_c}$, задающий алгоритм работы автомата и определяющий места подстановки элементов **a'** в шаблон; $\mathbf{I_{a_c}}$ – рецепторы $\mathbf{a_c}$; $\mathbf{O_{a_c}}$ – эффекторы $\mathbf{a_c}$; $\mathbf{\theta_{a_c}}$ – операция подстановки {**a'**}, заданного описаниями $\mathbf{D_{a_c}}$, в $\mathbf{h_{a_c}}$ вида $\mathbf{\theta_{a_c}:(h_{a_c},\{a'\})\rightarrow a_c}$.

Среди всех описаний автоматов выделим ***вырожденное описание*** вида $\mathbf{D(a_c^{simple})=}$ $\mathbf{=<null, D_{a_c^{simple}}, null, null, null>}$ или $\mathbf{D(a_c^{simple})=D_{a_c^{simple}}}$, где **null** – *пусто* (отсутствие), мощность множества $\mathbf{D_{a_c^{simple}}}$ равна единице, а входы и выходы автомата $\mathbf{a_c^{simple}}$ задаются входами и выходами единственного обособленного элемента **a'**.

Описание $\mathbf{D(a_c^{simple})}$, которое обозначим как $\mathbf{D_{a_c}^{simple}}$, задает ***простой автомат*** $\mathbf{a_c^{simple}}$.

Будем говорить, что описание $\mathbf{D(a_c)}$ ***допустимо*** **Cu** с реестром **R**, если каждое описание $\mathbf{d(a')\in D_{a_c}}$ допустимо. Описание **d(a')** допустимо, если для любого идентификатора из $\mathbf{Id_{a'}}$ в реестре существует автомат с таким идентификатором и если **a'** локален; это означает, что, если возбудить все рецепторы автомата **a'**, то хотя бы один эффектор перейдет в возбужденное состояние, то есть вершины $\mathbf{O_{a'}}$ графа **G** с матрицей смежности $\mathbf{Rel_{a'}}$ разрешимы относительно вершин $\mathbf{I_{a'}}$.

Для проверки первого условия допустимости **a' Cu** последовательно проверяет все элементы множества $\mathbf{Id_{a'}}$ с помощью операции ***contains*** на наличие требуемого автомата в реестре **R**.

Локальность конструируемого автомата проверяется по следующему алгоритму:

1) все вершины графа **G**, являющиеся входами автомата **a'**, отметить маркером *разрешимая*, остальные – *неразрешимая*;

2) если каждая вершина графа **G**, являющаяся выходом автомата **a'**, отмечена маркером *разрешимая*, то автомат **a'** локален и следует перейти к шагу 5;

3) последовательно просматриваем все *неразрешимые* вершины графа в поисках новых *разрешимых* вершин (*неразрешимая* вершина будет *разрешимой*, если все ее входы *разрешимы*);

4) если найдены новые *разрешимые* вершины, то отмечаем их маркером *разрешимая* и переходим к шагу 2, иначе граф **G** неразрешим и **a'**, а следовательно, и $\mathbf{a_c}$ нелокальны;

5) завершить алгоритм.

В качестве примера рассмотрим решение весьма тривиальной задачи: построить автомат, способный вычислять выражение **(a+b)*c**.

Описание автомата **(a+b)*c:**

$\mathbf{D^{simple}_{a_{(a+b)*c}}=<\{id_+, id_*\},}$

	X1	X2	X3	a	b	+	a+b	c	*	(a+b)*c	out
X1	0	0	0	1	0	0	0	0	0	0	0
X2	0	0	0	0	1	0	0	0	0	0	0
X3	0	0	0	0	0	0	0	1	0	0	0
a	0	0	0	0	0	1	0	0	0	0	0
b	0	0	0	0	0	1	0	0	0	0	0
+	0	0	0	0	0	0	1	0	0	0	0
a+b	0	0	0	0	0	0	0	0	1	0	0
c	0	0	0	0	0	0	0	0	1	0	0
*	0	0	0	0	0	0	0	0	0	1	0
(a+b)*c	0	0	0	0	0	0	0	0	0	0	1
out	0	0	0	0	0	0	0	0	0	0	0

$\mathbf{\{x_1,x_2,x_3\}, \{out\}>}$

Если описание $\mathbf{D(a_c)}$ допустимо, то **Cu** приступает к построению автомата $\mathbf{a_c}$, которое заключается в последовательном построении всех обособленных элементов и включении их в шаблон $\mathbf{h_{a_c}}$ с помощью операции $\mathbf{\theta_{a_c}}$. Отметим, что **Cu** не соединяет рецепторы и эффекторы конструируемого автомата $\mathbf{a_c}$ с рецепторами и эффекторами обособленных элементов, так как правила их взаимодействия определяются операцией $\mathbf{\theta_{a_c}}$ и шаблоном $\mathbf{h_{a_c}}$.

Для построения обособленного элемента **a'** автомата $\mathbf{a_c}$ **Cu** выполняет следующие действия:

– с помощью операции **get** извлекает из реестра **R** все необходимые элементы для автомата **a'**;

– по матрице $\mathbf{Rel_{a'}}$ соединяет рецепторы и эффекторы элементов **a'** между собой.

Построенный автомат **a'** с помощью операции $\mathbf{\theta_{a_c}}$ и шаблона $\mathbf{h_{a_c}}$ включается в автомат $\mathbf{a_c}$ путем соединения рецепторов и эффекторов автомата **a'** с эффекторами и рецепторами автомата $\mathbf{a_c}$. Такие соединения будем выполнять с помощью адаптеров.

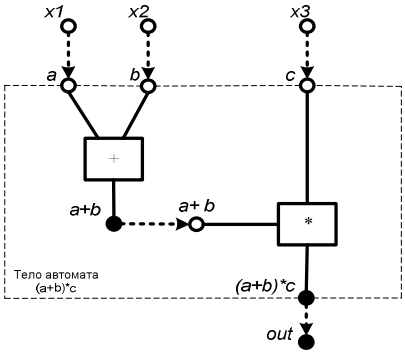
Адаптеры могут соединять рецепторы с рецепторами или рецепторы с эффекторами и в процессе передачи сигнала способны изменять его структуру. Алгоритм изменения сигналов **Cu** определяется на основе совместимости типов сигналов, которыми оперируют соединяемые адаптером элементы.

Предварительно сделаем аксиоматическое допущение, упрощающее процесс построения модели, от которого придется отказаться при решении практических задач: полагаем, что все передаваемые сигналы имеют одинаковую структуру и от-

личаются только своей мощностью. При этом роль адаптеров сводится лишь к передаче и согласованию входного и выходного сигналов.

Итак, после того как **Cu** осуществил все необходимые соединения, для созданного автомата $\mathbf{a_c}$ генерируется уникальный идентификатор, и новый автомат регистрируется в реестре **R** (в случае простого автомата $\mathbf{a_c^{simple}}$ регистрируется обособленный элемент **a'**).

Пример. Схематичное изображение автомата **(a+b)*c:**



Примечание: здесь и далее на схемах адаптер обозначен как ○⇢●.

Покажем, как в рамках предложенной модели реализуются ветвления и циклы. Для этого рассмотрим две задачи:

1) построить автомат $\mathbf{a_{sf}}$, реализующий кусочную функцию $\mathbf{f(a)}=\begin{cases}\mathbf{a}, & \mathbf{a\geq 0}\\ \mathbf{a*a}, & \mathbf{a<0}\end{cases}$;

2) построить автомат $\mathbf{a_{ff}}$, вычисляющий **a**-факториал **(a!)**.

Для построения требуемых автоматов необходимо наличие в реестре элементарного умножающего автомата ($\mathbf{a_*\in A_0}$). Автоматы $\mathbf{a_{sf}}$ и $\mathbf{a_{ff}}$ в предлагаемой модели можно построить двумя способами: с использованием вырожденного описания либо при помощи шаблонов $\mathbf{h_{sf}}$ и $\mathbf{h_{ff}}$ соответственно. В первом случае к реестру необходимо добавить автоматы: $\mathbf{a_{if-sf}}$, имеющий один вход **a**, два выхода ***l*** и **r** и работающий по правилу «если $\mathbf{a\geq 0}$, то подать сигнал **a** на выход **r**, иначе на выход ***l***»; $\mathbf{a_{if-ff}}$, имеющий два входа $\mathbf{a_i}$ и $\mathbf{res_i}$, три выхода $\mathbf{a_o}$, $\mathbf{res_o}$, **out** и работающий по алгоритму:

1) если $\mathbf{res_i}$ равен **null**, то на выход $\mathbf{res_o}$ подать сигнал $\mathbf{a_i}$, иначе – сигнал $\mathbf{res_i}$;

2) $\mathbf{a_i=a_i-1}$;

3) если $\mathbf{a_i}$ меньше **2**, то на выход **out** подать тот же сигнал, что и на выход $\mathbf{res_o}$, иначе на выход $\mathbf{a_o}$ подать сигнал $\mathbf{a_i}$;

4) конец.

Автоматы $\mathbf{a_{sf}}$ и $\mathbf{a_{ff}}$ будут иметь следующий вид:

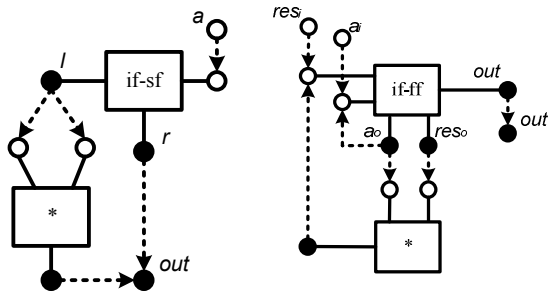


схема автомата $\mathbf{a_{sf}}$ *схема автомата* $\mathbf{a_{ff}}$

В случае использования шаблонов $\mathbf{h_{sf}}$ и $\mathbf{h_{ff}}$ автоматы можно представить следующим образом.

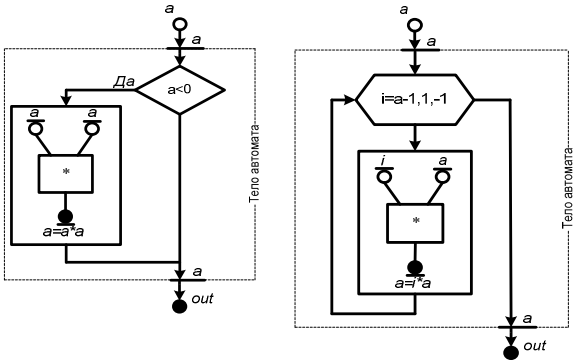


схема автомата $\mathbf{a_{sf}}$ *схема автомата* $\mathbf{a_{ff}}$

Таким образом, если в реестре КУА находятся все необходимые компоненты, он сможет построить любой корректно заданный автомат.

Предложенная модель КУА, несмотря на принятые допущения, имеет законченный вид и может использоваться для автоматической генерации программных систем.

Литература

Дж. фон Нейман. Теория самовоспроизводящихся автоматов. М.: Мир, 1971. 382 с.