ЗАДАНИЕ 4
Модель Рейнолдса

Построение модели • Выполним реализацию *двумерной* модели Рейнолдса, в которой птицы могут выполнять только два поворота — вправо или влево. Модель должна поддерживать три правила Рейнолдса: каждая птица должна по возможности избегать столкновений с другими птицами, выравнивать свою скорость согласно своим соседям и не отрываться от стаи. Пользователь должен иметь возможность отключать любое из правил для более полного понимания того, как они влияют на поведение всей стаи. Вместо использования приоритетов правил будем вычислять изменения скорости для каждого правила с последующим их рандомизированным усреднением.

A Создаем новую модель, размер мира устанавливаем равным 50 патчей по обоим измерениям, размер патча делаем равным 4 пикселям.

B К интерфейсу модели добавляем слайдер, привязанный к переменной `flock-size`, с диапазоном значений от 10 до 200, шагом 10 и единицей измерений «birds».

C Создаем новый вид черепах `birds`. Для этого в начале кода надо указать команду

```
breed [birds bird].
```

Двумя ее аргументами являются названия нового вида во множественном и единственном числе.

D Добавляем кнопку `setup`. Пишем код процедуры `setup`, в котором очищаем модель, раскрашиваем патчи в белый цвет, создаем стаю птиц числом `flock-size`. Каждую новую птицу помещаем

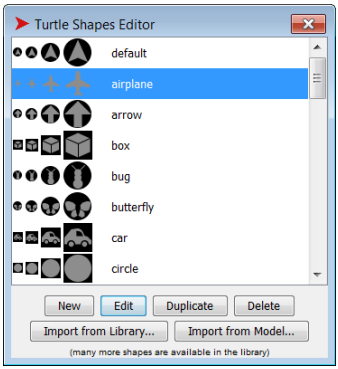

РИС. 4.1 Редактор форм

[1] Доступные в NetLogo формы черепах можно посмотреть через диалог Tools → Turtle Shapes Editor (рис. 4.1).

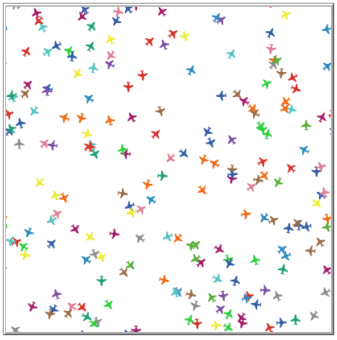

РИС. 4.2 Начальная конфигурация модели

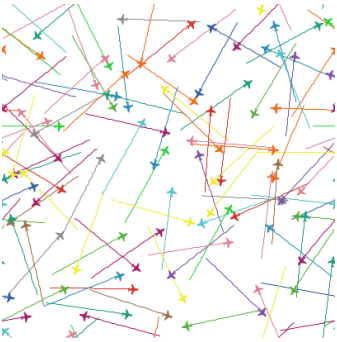

РИС. 4.3 Равномерное прямолинейное движение птиц (к модели была добавлена возможность включения/выключения режима трассировки)

в случайную точку пространства и устанавливаем для нее следующие параметры: размер — 3, форма — «airplane»[1]. Последней командой процедуры `setup` сбрасываем таймер. Проверяем работу кнопки `setup` (рис. 4.2).

E Добавляем слайдер `velocity` с диапазоном от 0.1 до 2.0, с шагом 0.1 и значением по умолчанию 0.7. Соответствующая переменная `bird-velocity` будет регулировать среднюю скорость птиц.

F Добавляем кнопку `go` и соответствующую процедуру, в которой:

- просим всех птиц обновить направление своей скорости (согласно правилам Рейнолдса) с помощью процедуры `update-heading`;
- передвигаем всех птиц на расстояние, определяемое средней скоростью с небольшим случайным возмущением:
  `fd velocity * random-normal 1.0 0.1`.
- обновляем таймер.

G Добавляем процедуру `update-heading`, в которой пока не выполняется никаких действий. Проверяем работу кнопки `go`. Птицы должны начать двигаться прямолинейно и относительно равномерно, никак не взаимодействуя друг с другом (рис. 4.3).

H Добавляем к интерфейсу два слайдера (в скобках указан диапазон, шаг изменения и значение по умолчанию):

- `vision` (1, 20, 0.5, 10) — радиус области контроля каждой птицы;
- `min-sep` (1, 10, 0.2, 3) — минимально допустимое расстояние до ближайшей птицы.

I Добавляем к интерфейсу три переключателя с именами `separate?`, `align?` и `flock?`. С их помощью можно будет включать и выключать соответствующие правила Рейнолдса, к реализации которых в процедуре `update-heading` мы и переходим.

J В начале процедуры `update-heading` определяем соседей (`mates`) данной птицы, которые находятся в области ее видимости (команда `in-radius`):

```
let mates other birds in-radius vision.
```

Если множество `mates` не пусто:

`if any? mates`,

то вычисляем новое направление движения с учетом включенных правил. В противном случае ничего менять в движении птицы не надо, т.к. вблизи нее нет ни одной другой птицы.

K Определяем три локальные переменные `alpha`, `beta` и `gamma`, которые задают веса каждого из трех правил. Значения этих переменных задаем случайным образом:

```
let alpha 0.2 + random-float 0.2
let beta 0.2 + random-float 0.2
let gamma 0.1 + random-float 0.1
```

L Создаем локальную переменную `h-sep`, которая будет задавать направление для уклонения от ближайшей птицы (если она находится в области видимости текущей птицы). Инициализируем эту переменную текущим направлением движения, которое хранится в атрибуте `heading`.

M Если включен режим `separate?`, то находим ближайшего соседа:

`let n min-one-of mates [distance myself]`.

Если расстояние до него меньше `min-sep`, то устанавливаем значение переменной `h-sep` равным

`180 + towards n`.

Команда `towards` возвращает направление (в градусах) на указанного агента. Прибавив к этому углу 180 градусов, мы получим угол направления *от* этого агента.

N Новое направление птицы будет средним (взвешенным) четырех единичных векторов — ее текущего направления и трех направлений, определяемых правилами Рейнолдса. Координаты единичного вектора текущего направления определяются атрибутами `dx` и `dy`. Учтем вклад первого правила в изменение направления:

```
let dx-new dx + alpha * sin h-sep
let dy-new dy + alpha * cos h-sep
```

Зная координаты вектора направления, его угол можно найти с помощью функции `atan`:

`set heading atan dx-new dy-new`.

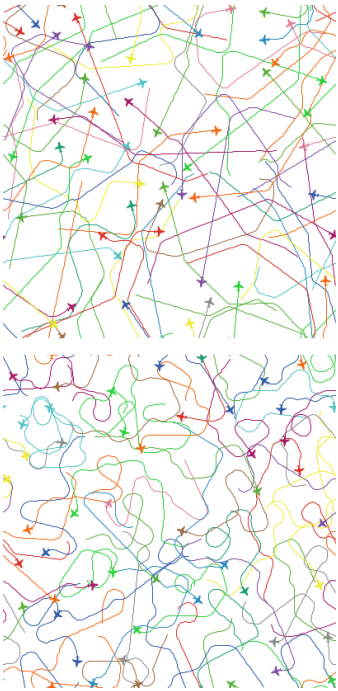

РИС. 4.4 Изменение характера движения птиц при увеличении параметра `min-sep` с 3 (вверху) до 10 (внизу)

Проверяем работу модели с включенным первым правилом и анализируем, как влияет изменение параметра `min-sep` на характер движения птиц (пример приведен на рис. 4.4).

O Переходим к реализации второго правила. По аналогии с первым правилом создаем локальную переменную `h-align`, которая будет задавать среднее направление движения птиц в области видимости текущей птицы. Инициализируем эту переменную текущим направлением.

P Если включен режим `align?`, то определяем новое направление `h-align`. Для этого находим покоординатную сумму всех единичных векторов, определяющих направления движения птиц из множества `mates`. После чего вычисляем само направление `h-align`.

```
let xa sum [dx] of mates
let ya sum [dy] of mates
set h-align atan xa ya
```

Команда `sum` вычисляет сумму заданного выражения для указанного набора агентов (короткий вариант команды `reduce +`).

Q Обновляем переменные `dx-new` и `dy-new` по аналогии с первым правилом, но используя весовой коэффициент `beta`.

```
set dx-new dx-new + beta * sin h-align
set dy-new dy-new + beta * cos h-align
```

Переносим команду вычисления нового направления `heading` после данного фрагмента кода. Проверяем работу второго правила Рейнолдса (рис. 4.5).

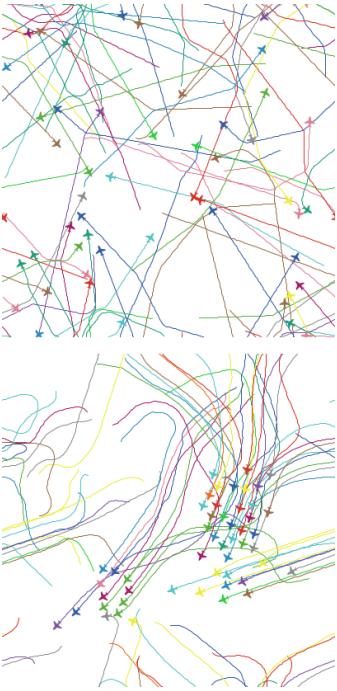

РИС. 4.5 Эффект включения второго правила (вверху правило выравнивания скоростей выключено, внизу — включено)

R Завершим построение модели реализацией последнего (третьего) правила Рейнолдса. Создаем локальную переменную `h-flock`, которая будет показывать среднее направление ко всем птицам в области видимости. Полагаем значение этой переменной равной текущему направлению.

S Если включен режим `flock?`, то определяем новое значение направления `h-flock`. Т.к. наша модель использует циклические границы, то просто найти среднее положение птиц из `mates` нельзя. Вместо этого мы найдем среднее направление на птиц в области видимости. Для этого определяем

средние значения направлений (отдельно по горизонтали и по вертикали) на всех птиц. После этого вычисляем само направление.

```
let xf sum [sin (towards myself + 180)] of mates
let yf sum [cos (towards myself + 180)] of mates
set h-flock atan xf yf
```

Команда `towards myself` вычисляет направление от некоторой птицы из набора `mates` к текущей птице. К этому направлению (это угол) мы прибавляем 180 (градусов), чтобы получить обратное направление.

T Обновляем переменные `dx-new` и `dy-new` для направления `h-flock` и веса `gamma`. Проверяем работу третьего правила Рейнолдса (рис. 4.6).

U Проверяем работу модели, ее окончательный интерфейс показан на рис. 4.7. Для этого тестируем различные ее режимы, включая и выключая правила Рейнолдса, пробуем различные значения параметров модели.

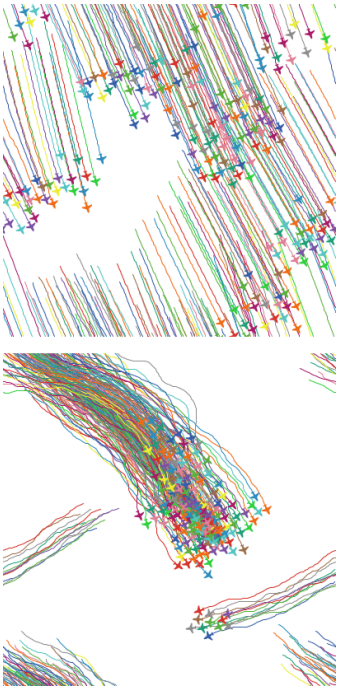

РИС. 4.6 Траектории движения стаи птиц с выключенным (вверху) и включенным (внизу) режимами `flock?` (при включенных первых двух правилах)

УПРАЖНЕНИЯ

1 Включите в модель режим трассировки, который будет наглядно показывать траектории движения всех птиц в стае.

2 Добавьте к интерфейсу модели слайдеры для динамического изменения различных параметров модели (например, пределов изменения весов `alpha`, `beta` и `gamma`). Исследуйте, как меняется поведение стаи птиц при динамическом изменении этих параметров. Примеры роевого поведения для экстремально больших (из диапазона от 2 до 7) значений указанных параметров показаны на рис. 4.8.

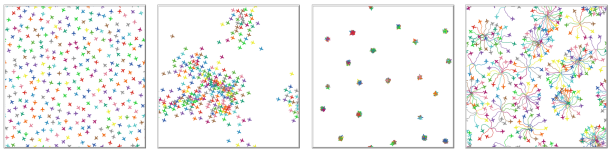

РИС. 4.8 Различные типы роевого поведения в модели

3 Добавьте к интерфейсу модели три графика, демонстрирующих количественно аспекты роевого поведения, обусловленные применением каждого из трех правил Рейнолдса (рис. 4.9).

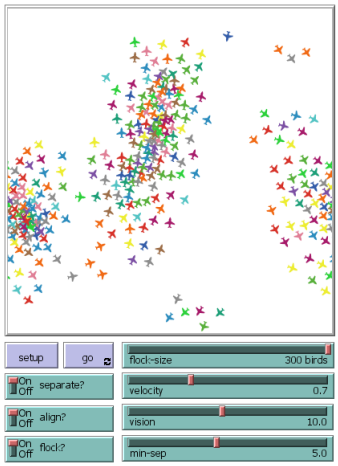

РИС. 4.7 Окончательный интерфейс модели

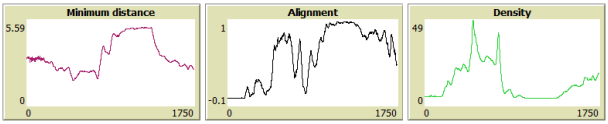

РИС. 4.9 Графики, демонстрирующие эффект применения правил Рейнолдса

Все графики должны показывать усредненные по всем птицам следующие характеристики. Для первого правила — расстояние до ближайшего соседа. Для второго правила — средняя степень выравненности скорости птицы относительно остальных птиц. Эту степень для двух птиц можно посчитать как скалярное произведение единичных векторов их скоростей (атрибуты `dx` и `dy`) — фактически это косинус угла между двумя векторами. Для третьего — число птиц в окрестности радиуса `vision`.

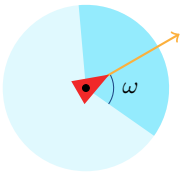

РИС. 4.10 Область контроля с ограниченным сектором размера ω градусов

4 При полете птицы большее внимание должны уделять поведению своих соседок по стае, которые находятся впереди них, а не сзади. Модифицируйте модель так, чтобы соседними (`mates`) для каждой птицы были только те птицы, которые находятся внутри сектора круга радиуса `vision` с углом ω в обе стороны относительно направления данной птицы (рис. 4.10). Для этого используйте встроенную команду `in-cone` вместо команды `in-radius`. Эта команда имеет два аргумента — радиус видимости и угол 2ω (полный сектор). На рис. 4.11 показаны области контроля птицы, выделенной красным цветом, для исходной модели и для модели с ограниченным сектором размера $\omega = 90$ градусов. Протестируйте сделанные изменения в модели, установив угол видимости $\omega = 90$. Интересно, что в такой конфигурации даже с отключенным выравниванием скорости (второе правило) стая птиц показывает скоординированное (двустороннее!) движение (рис. 4.12) вдоль некоторой криволинейной траектории.

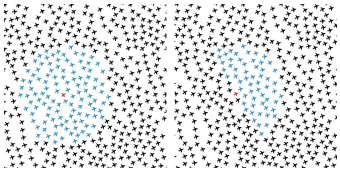

РИС. 4.11 Области контроля для исходной модели (слева) и для модели с ограниченным сектором видимости (справа)

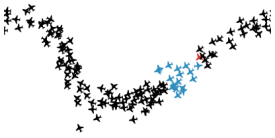

РИС. 4.12 Пример самоорганизации птиц в модели с ограниченным сектором видимости

5 Исследуйте, как изменится поведение стаи птиц, если каждая птица будет координировать свое движение только с заданным числом (например, тремя) ближайших к ней птиц. Таким образом, радиус области видимости каждой птицы оказывается динамическим и разным для разных птиц. Чтобы реализовать такую вариацию модели, множество `mates` надо вычислять командой

```
min-n-of 3 other birds [distance myself].
```

Первым аргументом команды `min-n-of` является число отбираемых агентов, вторым аргументом — исход-

ный набор агентов, из которого делается отбор, третьим — функция, которая используется для упорядочивания агентов.

6 Включите в модель учет разного рода неподвижных препятствий. Для этого припишите патчам атрибут `obstacle?`, который будет указывать, является ли данный патч препятствием или нет. Раскрасьте препятствия черным цветом. Уклонение от препятствий должно быть более приоритетным, чем все остальные правила. Поэтому обнаружение препятствий и реагирование на них нужно выполнять до реализации правил Рейнолдса (в процедуре `update-heading`). Обнаружение препятствий можно реализовать по следующей схеме. Находим число препятствий в правом секторе обзора (поворачиваемся направо, считаем препятствия, поворачиваем обратно), потом в левом секторе. Если препятствия есть (либо справа, либо слева), то поворачиваем на небольшой случайный угол в ту сторону, где препятствий меньше. Если их одинаковое количество, то направление поворота выбираем случайным образом. Если препятствий нет, то переходим к правилам Рейнолдса. Для начала объявите препятствием единственный центральный патч модели. Посмотрите, как стая птиц разделяется на две части, если препятствие оказывается на ее пути (рис. 4.13).

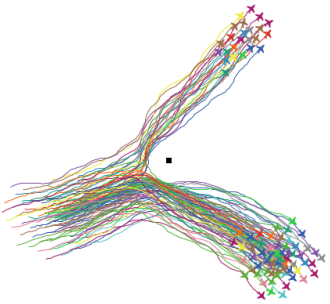

РИС. 4.13 Разделение стаи при встрече с препятствием

7 Рассмотрите случай нескольких препятствий, расставленных по полю модели случайным образом. Пример с десятью препятствиями приведен на рис. 4.14, на котором показаны траектории движения стаи птиц.

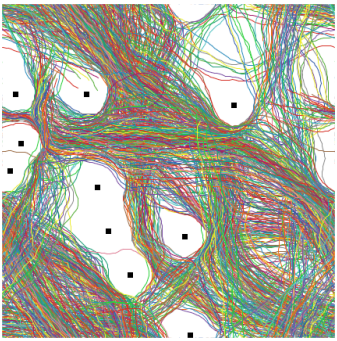

РИС. 4.14 Случай нескольких препятствий

8 Постройте модель, в которой препятствия образуют стены, ограничивающие область полета птиц. В простейшем случае стенами служат патчи с минимальными и максимальными координатами по вертикали и по горизонтали (рис. 4.15).

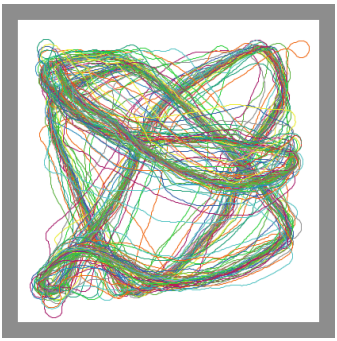

РИС. 4.15 Движение стаи птиц в ограниченном пространстве

9 Реализуйте модель с хищником. Для этого добавьте в модель новый вид `predator`, определите его визуальные характеристики и способ его передвижения (например, фиксированная скорость с небольшим случайным изменением направления на каждом временном шаге). С точки зрения птиц хищник должен быть препятствием, которое надо облетать. Поэтому поведение птиц может быть таким же, как и в предыдущих трех упражнениях (с заменой препятствия-патча на хищника). Исследуйте поведение стаи при наличии хищника (рис. 4.16). Интересно, что при большом числе птиц и выключенном режиме `flocking?` их движение с некоторого момента времени начинает полностью опре-

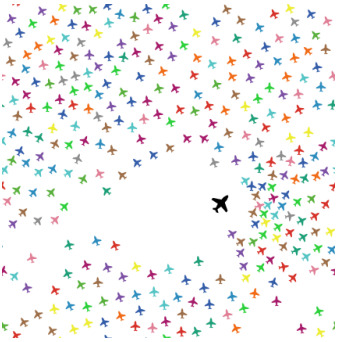

РИС. 4.16 Модель с хищником

[2] T. Vicsek, et al., *Novel Type of Phase Transition in a System of Self-Driven Particles*, Phys. Rev. Lett. 1995, 75, 6, p. 1226–1229.

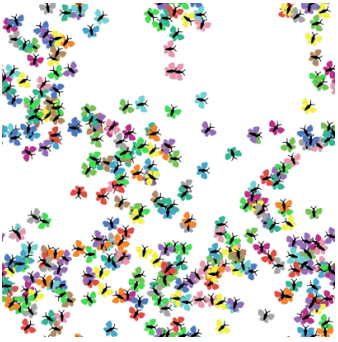

РИС. 4.17 Модель SPP

[3] T. Schmickl, et al., *How a life-like system emerges from a simple particle motion law*, Sci. Rep. 2016, 6:37969.

деляться направлением движения хищника.

10 Более простой моделью, в которой возникает согласованное движение отдельных частиц, является модель SPP (self-propelled particles)[2]. В этой модели частицы имеют фиксированную скорость, направление θ_t которой для каждой частицы пересчитывается согласно следующей формуле:

$$\theta_{t+1} = \langle\theta_t\rangle_r + \triangle\theta, \tag{4.1}$$

где $\langle\theta_t\rangle_r$ — среднее направление скорости всех частиц из локальной окрестности размера r данной частицы (включая ее саму), $\triangle\theta$ — случайный угол из диапазона $[-\eta/2, \eta/2]$. Реализуйте такую модель и проанализируйте ее поведение. Заметим, что движение частиц в модели SPP оказывается непохожим на поведение птиц, а больше напоминает движение бабочек или мотыльков. Попробуйте использовать форму частиц `"butterfly"` (рис. 4.17).

11 В 2016 году была предложена вариация модели SPP[3]. В этой модели используется другой алгоритм вычисления нового направления скорости. Каждая частица определяет число L и R соседей в левой и правой частях (относительно своего направления) своей локальной окрестности радиуса r. Новое направление тогда задается формулой

$$\theta_{t+1} = \theta_t + \alpha + \beta \cdot (R + L) \cdot \mathrm{sign}(R - L), \tag{4.2}$$

где α и β — параметры модели. Т.е. степень поворота частицы зависит от того, во сколько раз больше у нее соседей справа или слева относительно текущего ее направления. Интересным свойством данной модели (для случая $\alpha = 180°$, $\beta = 17°$, $r = 5$ и скорости частиц $v = 0.67$) является возникновение в ней кольцевых структур, обладающих способностью роста и деления (рис. 4.18). Для визуализации этого процесса частицы раскрашиваются в зависимости от числа N своих соседей в окрестности заданного радиуса r: если $N \in [16, 35]$, то цвет красный, иначе — серый.

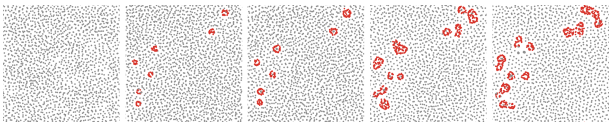

РИС. 4.18 Рост и деление клеточных структур в вариации модели SPP