

Статья. Библиографические данные:

Житников А.П.
Алгоритмический анализ параллельных Scratch-программ.
// Современные технологии преподавания естественно-научных дисциплин
в системе общего и профессионального образования.
Сб. матер. Междунар. науч.-практ. форума.
– Борисоглебск: БГПИ, 2012. – С. 66-79.

АЛГОРИТМИЧЕСКИЙ АНАЛИЗ ПАРАЛЛЕЛЬНЫХ SCRATCH-ПРОГРАММ (детская параллельная алгоритмика)

Статья приводится с технической доработкой:
гиперссылки, оглавление, цветовые элементы и т.п.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ПРИМЕР 1. ПРОСТАЯ АНИМАЦИЯ "PLAYGROUND"	4
1.1 Исходные данные.....	4
Объект алгоритмического анализа	4
Общий анализ работы программы.....	4
Постановка алгоритмической задачи	5
1.2 Обобщенное описание алгоритма сценария.....	6
Исходные литерные обозначения.....	6
Структурная формула общего алгоритма сценария	6
Структурная схема общего алгоритма сценария.....	6
1.3 Обобщенное описание алгоритма управления сценарием.....	8
Дополнительный анализ модели.....	8
Дополнительные сведения.....	8
Дополнительные исходные литерные обозначения.....	8
Структурная схема алгоритма управления сценарием	8
Структурная формула алгоритма управления сценарием	9
Исполнение алгоритма во времени	10
Проблема отображения внешних команд start и stop	11
Обобщенный алгоритм (надалгоритм) управления системой моделирования	11
1.4 Конкретизация алгоритма по программным скриптам	12
Уточнение постановки алгоритмической задачи	12
Диаграммы Насси-Шнейдермана	12
Уточнение структурной схемы общего алгоритма	13

Уточнение структурной формулы общего алгоритма	13
Анализ полученных структурных формул	14
1.5 Текущие итоги	14
2 ПРИМЕР 2. ПРОСТАЯ ИГРОВАЯ АНИМАЦИЯ "FISHCHOMP"	16
2.1 Исходные данные	16
Объект алгоритмического анализа	16
Общий анализ работы программы	16
Постановка алгоритмической задачи	16
2.2 Обобщенное описание алгоритма управления сценарием	18
Исходные литерные обозначения	18
Структурная схема общего алгоритма управления сценарием	18
Структурная формула алгоритма управления сценарием	18
ЗАКЛЮЧЕНИЕ	20
ЛИТЕРАТУРА	21
ПРИЛОЖЕНИЕ А Дополнение статьи: продолжение анализа Примера 2	22

ВВЕДЕНИЕ

К настоящему времени сложилась обширная *массовая международная практика детского параллельного программирования* на основе учебного языка визуального программирования *Scratch (Скретч)*. Данный язык программирования является одним из направлений развития известного учебного языка программирования Logo в области *детского компьютерного творчества*. Это направление детского компьютерного творчества организуется в рамках международного проекта на основе одноименного *многоязычного* (англоязычного, русскоязычного и т.п.) *учебного конструктора* мультимедийных и интерактивных игр *Scratch* [1-3].

Представляет исключительно большой интерес *алгоритмический анализ* такой реально существующей массовой практики детского параллельного программирования:

- что оно собой представляет с точки зрения теории параллельных алгоритмов?
- какие выводы и действия могут последовать из такого анализа?

Эта задача имеет и более широкое значение, поскольку среда программирования Scratch применяется также *студентами* и *преподавателями* для разработки параллельных мультимедийных моделей разного типа.

В статье кратко представлен первичный опыт алгоритмического анализа "параллельного" детского творчества такого типа.

1 ПРИМЕР 1. ПРОСТАЯ АНИМАЦИЯ "PLAYGROUND"

1.1 Исходные данные

Объект алгоритмического анализа

Ниже (Рис. 1.1) приводится окно сцены среды программирования Scratch с очень простой загруженной программой Playground.

Это **замечательная** и **показательная** во многих отношениях программа типа "**детский примитив**" с самостоятельными параллельными каракулями. И не случайно эта программа – пример № 1 в стандартном комплекте примеров (проектов) этой среды в разделе Animation (Анимация).

"Техническая" модельная задача PlayGround (Детская площадка)	Содержание сцены программы
	<p>На сцене представлены два действующих графических объекта – два спрайта:</p> <p>swing: качели 1 – подвесные веревочные качели (на наклонных опорах);</p> <p>seesaw: качели 2 – доска для качания (на треугольной подставке).</p> <p>Программа, по-видимому, не закончена. Но, это полезно – демонстрация минимального параллелизма (степени два).</p>

Рис. 1.1. Окно сцены модельной среды Scratch с программой Playground

Содержание параллельной анимации предельно простое (что представляет большой интерес с учебно-методической точки зрения на начальных этапах обучения параллельным представлениям):

при запуске программы (кнопкой Пуск: Start) начинают качаться веревочные качели и доска на подставке (выполняются два периодических параллельных процесса) – неопределенно долго до останова программы (кнопкой Останов: Stop).

Общий анализ работы программы

После запуска программы оба модельных объекта очень упрощенно имитируют **два независимых периодических механических процесса** (Рис. 1.2),:

- качели 2 (seesaw) имеют 2 крайние позиции, которые поочередно сменяются через заданную величину задержки во времени;
- качели 1 (swing) имеют 4 позиции (2 крайние позиции и по 1-й промежуточной позиции в каждом направлении движения), которые также поочередно сменяются через заданную величину задержки.

В каждой позиции спрайты меняют свои формы – костюмы.

Заданы задержки длительностью 1 сек., но их можно независимо изменять.

Два непрерывных периодических физических процесса заменяются их "грубыми" периодическими дискретными моделями. Однако достаточно понятно и приемлемо визуально имитируются два независимых параллельно (одновременно) выполняемых периодических процесса:

механические колебания – качание двух интуитивно хорошо понятных механических объектов (из области раннего детского жизненного опыта).

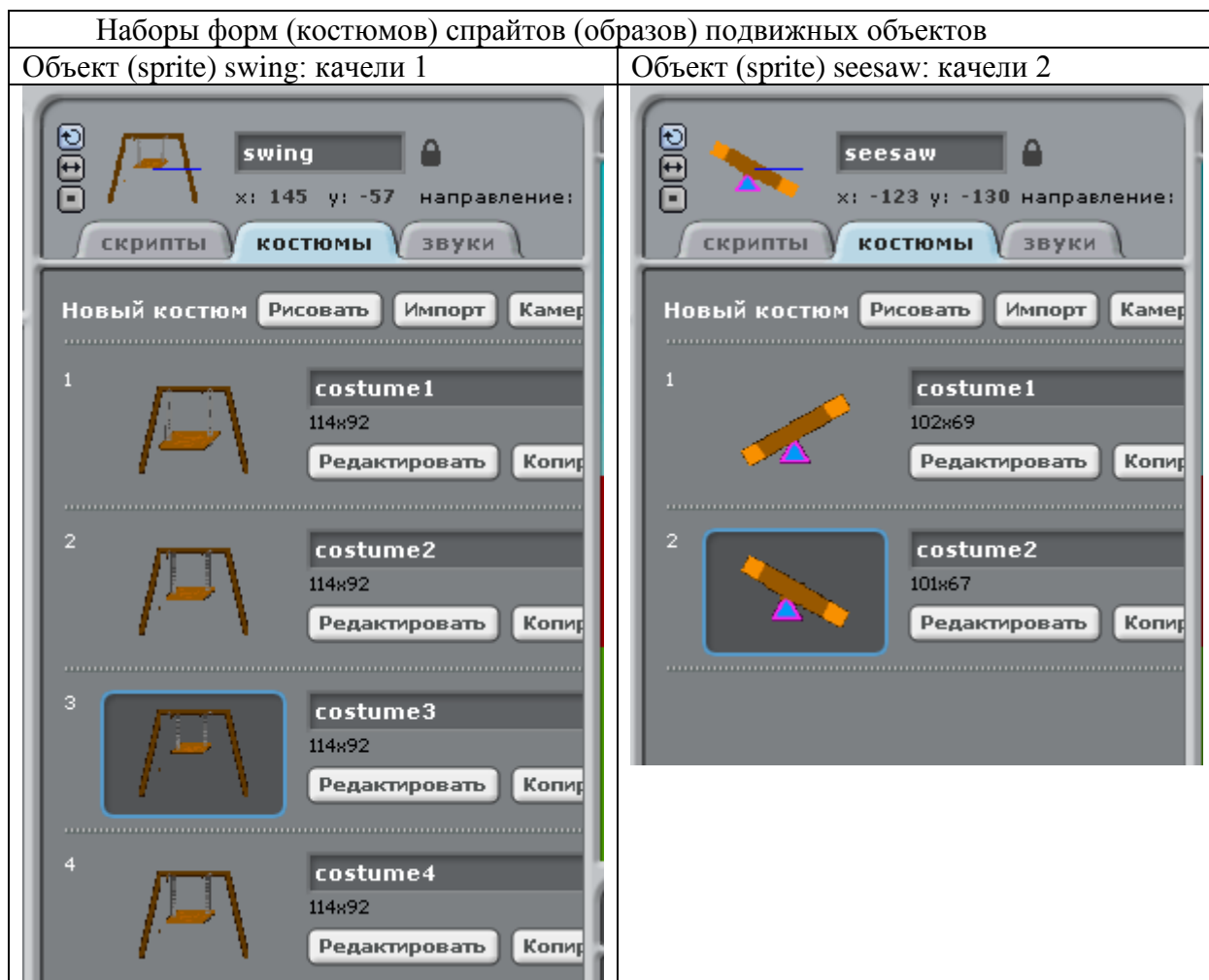


Рис. 1.2. Модельные позиции модельных объектов (спрайтов)

Постановка алгоритмической задачи

Принимаются следующие исходные условия.

Дано:

- 1) Программа PlayGround как объект алгоритмического анализа:
с наличием скриптов (данных и подпрограмм) действующих в программе объектов – приводятся далее.
- 2) Краткое описание ее работы (представленное выше).

Надо. Представить поэтапное описание алгоритма, реализуемого программой PlayGround, включая этапы анализа:

1) Общее алгоритмическое описание задачи – по визуальному анализу работы программы (независимо от программной реализации алгоритма), включая этапы:

- 1.1) Обобщенное описание **алгоритма сценария** задачи.
- 1.2) Обобщенное описание **алгоритма управления сценарием**.
- 1.3) Обобщенное описание **алгоритма управления модельной программой** – алгоритма пользователя программы
(в данной статье пока не реализовано, но этот вопрос обсуждается).

2) **Конкретизация алгоритма** по программным скриптам объектов:
адаптация алгоритма к парадигме и языку программирования.

1.2 Обобщенное описание алгоритма сценария

Исходные литерные обозначения

Принимаются (Табл. 1.1) условные обозначения общего оператора алгоритма и составляющих его операторов алгоритмов двух объектов.

Табл. 1.1. Исходные литерные обозначения

ЛТА: Литерный текст алгоритма	// ЛЗФ: Литерная знаковая форма
УЛО: Условные литерные обозначения	
ОА: Общее обозначение (оператора) алгоритма	
A100: swaying: качание: моделирование колебательных процессов – качание объектов типа качели.	
КСА: Комплект составляющих (операторов) алгоритмов	
A1: ПП1: Периодический процесс 1: swing: качание подвесных качелей: колебательный периодический процесс объекта swing	
A2: ПП2: Периодический процесс 2: seesaw: качание доски: колебательный периодический процесс объекта seesaw	

Структурная формула общего алгоритма сценария

В следующей таблице (Табл. 1.2) представлено литерное описание алгоритма типа комплекта двух операторов (структурная формула – формула состава в данном случае).

Табл. 1.2. Литерное описание алгоритма

ЛТА: Литерный текст алгоритма	// ЛЗФ: Литерная знаковая форма
СФА: Структурная формула алгоритма =	
КСО: Комплект составляющих операторов	
ОМФ: Одномерная форма / ГК: Горизонтальная компоновка Times: Шрифт типа Times – с переменным шагом	
$A100 = (A1, A2) = (swing, seesaw) = (\rightarrow A1 \rightarrow, \rightarrow A2 \rightarrow)$	
ДМФ: Двухмерная форма / ВК: Вертикальная компоновка Courier: Шрифт типа Courier – с постоянным шагом	
$A100 = (A1, A2) = (A1 \downarrow A2) = (A1) = (\rightarrow A1 \rightarrow) = \rightarrow (\rightarrow A1 \rightarrow) \rightarrow$ $(A2) \quad (\rightarrow A2 \rightarrow) \quad \rightarrow (\rightarrow A2 \rightarrow) \rightarrow$	
Символ "↓" (перевод строки) – условное обозначение (компоновочной) операции внутрискобочного перевода строки	

Общий алгоритм $A100 = (A1, A2)$ представляет собой простой комплект (кортеж записи) двух составляющих его алгоритмов A1, A2 управления моделями двух действующих объектов сценария. Они определяют два независимых (периодических) процесса. Связи операторов отсутствуют – это **вырожденная структура**: бесструктурный алгоритм – без связей компонент состава общего составного алгоритма (пустое множество связей).

Структурная схема общего алгоритма сценария

Далее (Рис. 1.3) приводится структурная схема алгоритма, выполненная средствами векторной графики в двух формах ее представления и компоновочного исполнения (по направлениям потока управления):

блок-схема и производная от нее компактная штрих-схема алгоритма.

Очевидно **прямое структурное соответствие** двухмерной структурной формула (Табл. 1.2) и двухмерной структурной схемы алгоритма: прямое совпадение их структур (наложением) с точностью до переобозначений (предельный случай изоморфизма – возможно с некоторым масштабированием)).

При этом **схемоподобная** структурная формула может рассматриваться как **литерная псевдографика**, которая является **шаблоном** для построения структурной схемы в **векторной графике** (и наоборот).

Двухмерная форма (ДМФ) структурной формулы алгоритма (Табл. 1.2, СФА/ДМФ) и обе структурные схемы алгоритма (ССА = БСА, ШСА) наглядно отражают наличие двух независимых **потоков исполнения** или **потоков управления** общего комплекта алгоритмов А100.

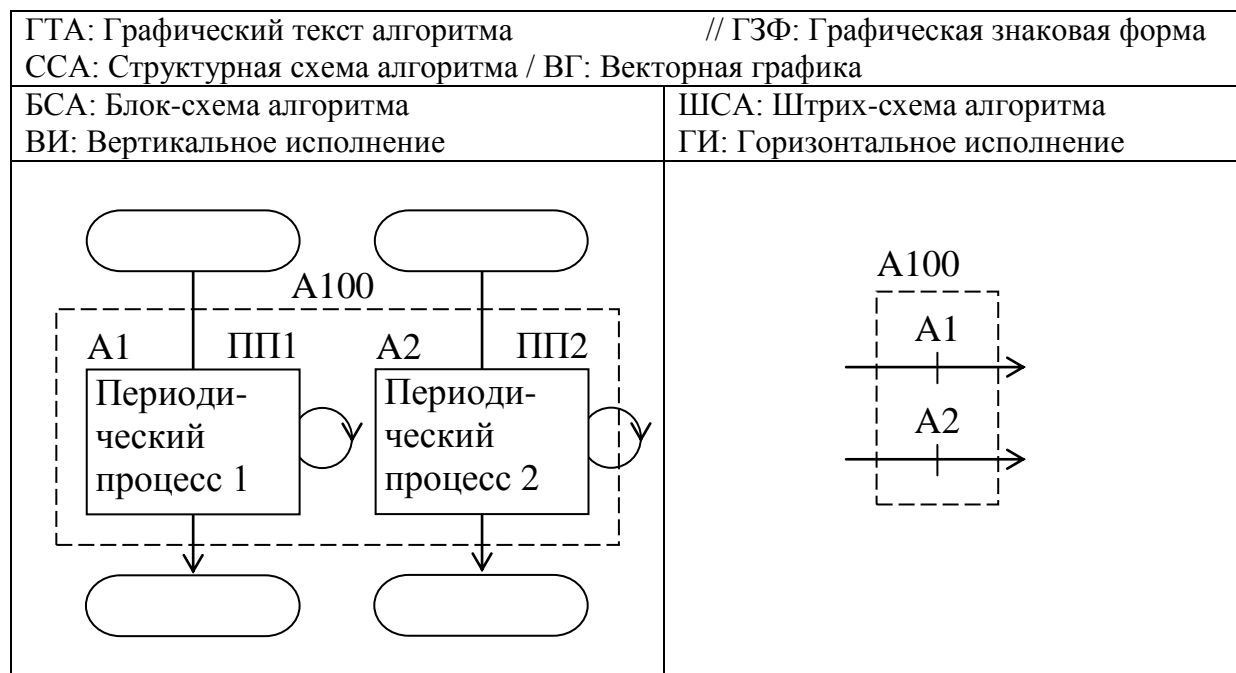


Рис. 1.3. Общая (обобщенная) схема алгоритма сценария

1.3 Обобщенное описание алгоритма управления сценарием

Дополнительный анализ модели

Независимые друг от друга составляющие алгоритмы (процессы) A1 и A2 комплекта алгоритмов $A100 = (A1, A2)$ в принципе могут запускаться в любое время. Можно наложить структурные операции связи, например:

$\rightarrow(A1, A2) = (A1 \rightarrow A2)$ – последовательная связь операторов;

$\|(A1, A2) = (A1 \parallel A2) = (A1 \# \circ A2) = \#(A1, A2) \circ$ – их параллельная связь.

Последовательная связь (\rightarrow : **секвенция**) задает последовательное выполнение операторов во времени, что в данном случае не подходит.

Параллельная связь (\parallel : **параллель**) задает параллельное во времени (одновременное) выполнение составляющих операторов. Это в принципе могло бы подойти, причем параллель $\parallel = \# \circ$ – это парная операция, где:

- знак # (диз) представляет **дивергенцию (разделение, распараллеливание)** потоков управления;
- знак \circ (круг) представляет **конвергенцию (соединение)** параллельных потоков управления.

Данная система подходит по начальным условиям работы модельной системы сценария (одновременный запуск периодических процессов), но не соответствует факту неопределенно большой их продолжительности (до общего прекращения работы модельной системы). В данном случае может быть использована только одна структурная операция дивергенции:

$\#(A1, A2) = (A1 \# A2)$ – распараллеливание двух потоков (Рис. 1.4).

Дополнительные сведения

Структурные формулы представлены в разных **синтаксических формах** записи структурных операций:

- **префиксная** форма (ПрФ):

$\rightarrow(A1, A2), \|(A1, A2), \#(A1, A2);$

- **инфиксная** форма (ИнФ):

$(A1 \rightarrow A2), (A1 \parallel A2) = (A1 \# \circ A2), (A1 \# A2);$

- **комбинированная** форма (КоФ):

$\#(A1, A2) \circ$ – в данном случае префиксно-постфиксная форма.

Описание исходных принципов построения языка структурных формул приводится в статьях [4,5]. Это является расширением классического языка логических схем алгоритмов (ЛСА, особых структурных формул алгоритмов) и его расширения на параллельные ЛСА (ПЛСА). Данные статьи приводятся на сайте **paralg1000.narod.ru** (или **paralg.narod.ru**) – теперь **paralg.ucoz.com**.

Дополнительные исходные литерные обозначения

Принимаются (Табл. 1.3) дополнительные обозначения операторов общего алгоритма управления A200 и начального алгоритма A0.

Структурная схема алгоритма управления сценарием

Далее (Рис. 1.4) приводится структурная схема общего алгоритма A200 управления сценарием алгоритма. Данная схема включает в себя:

- начальный блок A0 подготовки работы сценария;

- комплект A100 = (A1, A2) основных алгоритмов моделирования двух действующих объектов сценария, подключенный к начальному объекту посредством узла вилки # (fork), выполняющего функцию операции дивергенции: разделения начального потока управления на два параллельных (одновременно выполняемых) составляющих потока.

Табл. 1.3. Дополнительные исходные литерные обозначения

ООА: Общее обозначение (оператора) алгоритма
A200: управление: общее управление сценарием: качание объектов типа качели.
СКА: Система (операторов) команд алгоритма – дополнительно:
A0: ПС: Подготовка сценария – подготовка начало сеанса моделирования: запуск программы и подготовка выполнения параллельных процессов

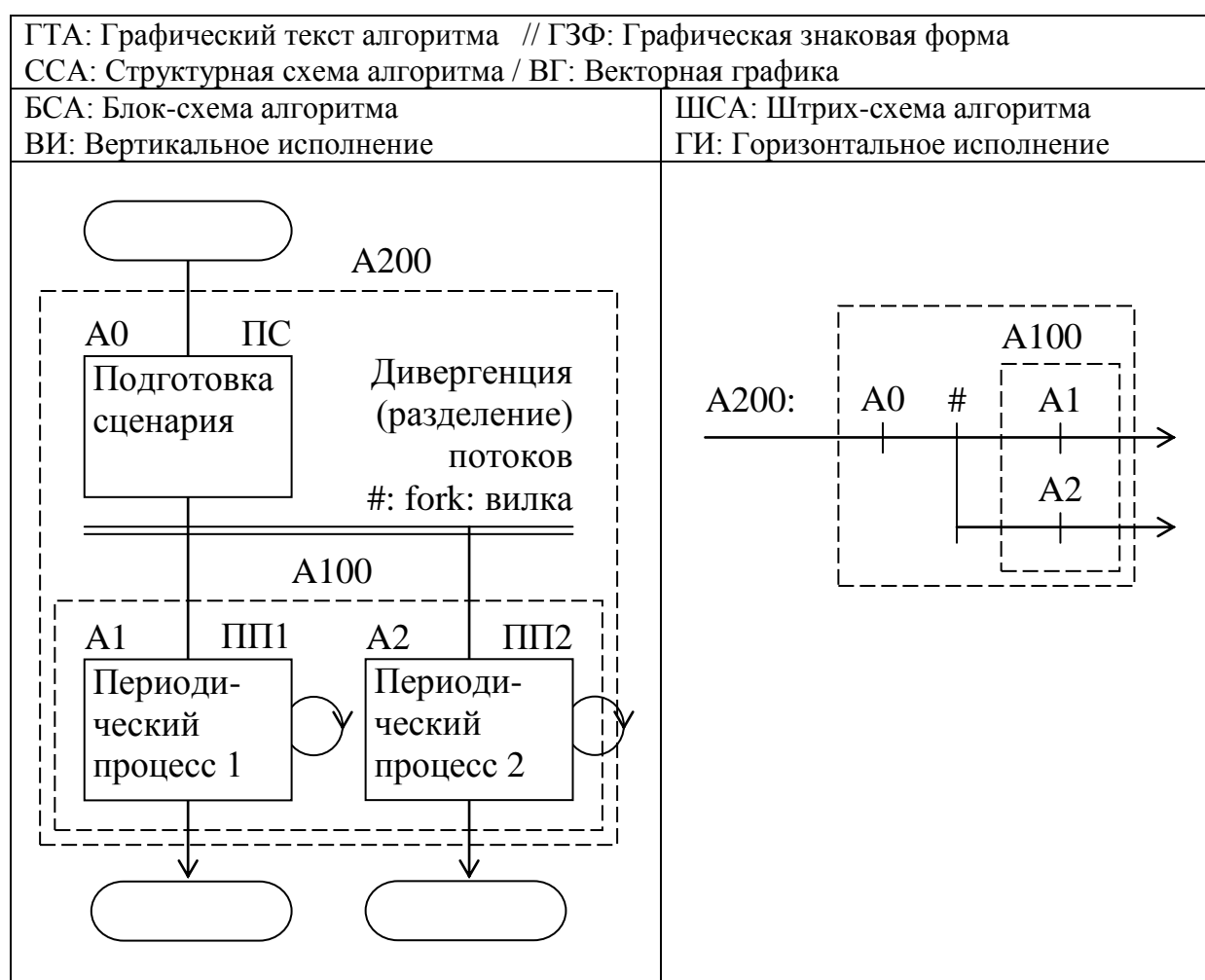


Рис. 1.4. Общая схема алгоритма управления сценарием

Структурная формула алгоритма управления сценарием

Далее (Табл. 1.4) приводится структурная формула алгоритма, определяемая согласно принятой структурной схеме алгоритма. Формула приводится в двух формах представления:

одномерная форма (СФА/ОМФ) и производная от нее двухмерная форма представления (СФА/ДМФ).

При этом **схемоподобная** структурная формула, как и ранее, может рассматриваться как **литерная псевдографика**, которая является **шаблоном** для построения структурной схемы в **векторной графике** (и наоборот).

Табл. 1.4. Литерное описание алгоритма

ЛТА: Литерный текст алгоритма // ЛЗФ: Литерная знаковая форма
СФА: Структурная формула алгоритма
ОМФ: Одномерная форма / ГК: Горизонтальная компоновка
$A200 = \rightarrow(A0, \#A100)) = \rightarrow(A0, \#(A1, A2)) =$ $= (A0 \rightarrow \#(A1, A2)) = (A0 \rightarrow (A1 \# A2))$ $A200 = (A0 \rightarrow \#(A1, A2)) = (\rightarrow A0 \rightarrow \#(\rightarrow A1 \rightarrow, \rightarrow A2 \rightarrow))$
ДМФ: Двухмерная форма / ВК: Вертикальная компоновка
$A200 = (A0 \rightarrow \#(A1, A2)) = (A0 \rightarrow \#(A1 \downarrow A2)) = (A0 \rightarrow \#(A1)) =$ $(A2)) =$ $= (\rightarrow A0 \rightarrow \#(\rightarrow A1 \rightarrow)) = (\rightarrow A0 \rightarrow \# (\rightarrow A1 \rightarrow)) = \rightarrow A0 \rightarrow \# \rightarrow A1 \rightarrow$ $(\rightarrow A2 \rightarrow) \quad ((\rightarrow A2 \rightarrow) \quad \rightarrow A2 \rightarrow$

Исполнение алгоритма во времени

Далее (Рис. 1.5) представлена диаграмма исполнения алгоритма в форме временной диаграммы, масштабированной по шкале времени.

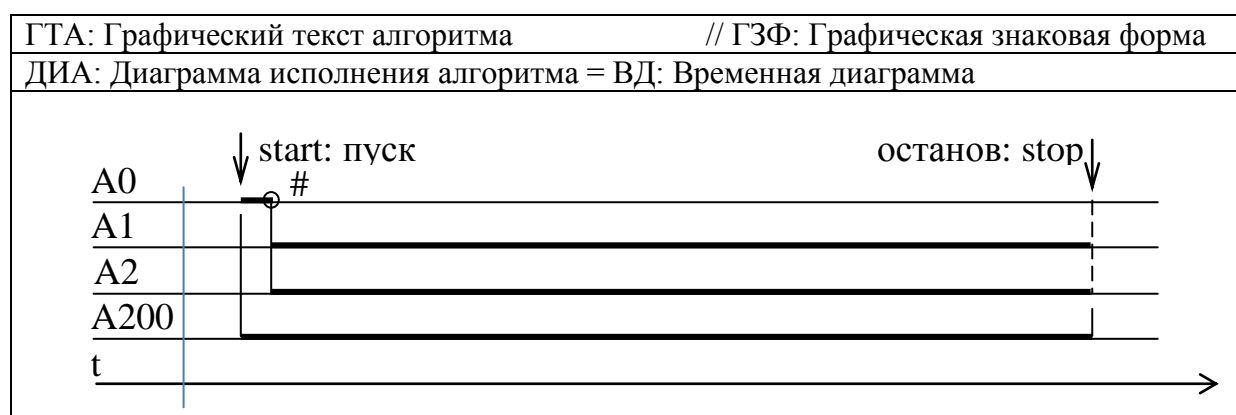


Рис. 1.5. Временная диаграмма исполнения алгоритма

Временная диаграмма четко отражает в визуальной форме следующий порядок действий, предписанный алгоритмом:

- после подачи внешнего сигнала пуск (start) от (внешнего) пользователя программы выполняется начальный оператор A0 запуска программы и подготовки системы;
- после его выполнения через узел вилки # (fork) одновременно (или почти одновременно) запускаются операторы A1, A2 двух процессов, которые далее выполняются самостоятельно и независимо друг от друга;
- через некоторое время (внешний) пользователь включает кнопку останова (stop) – этот внешний сигнал поступает в систему моделирования и программа отключается (сеанс моделирования прекращается).

Проблема отображения внешних команд start и stop

Существует концептуальная проблема привязки внешних сигналов **пуска** и **останова** (**start** и **stop**) алгоритма к схеме алгоритма (Рис. 1.6).

При этом:

- сигнал **пуска** (start) четко локализуется на схеме алгоритма: он **подается на вход** (схемы) алгоритма, но его **источник не принадлежит** алгоритму;
- однако сигнал **останова** (stop) не имеет четкой локализации в алгоритме: он запускает механизм прерывания (без возобновления) хода выполнения общего алгоритма управления A200 (это функция операционной системы);
- его **источник также не принадлежит** алгоритму, и он может быть приложен **в любом сечении** общей схемы процесса выполнения составного оператора A200: практически **в любое время** его выполнения.

Фактически неявно представлен **алгоритм пользователя** – это некоторый **надалгоритм** (A300) применения алгоритма управления A200.

Представляет интерес проработка этого общего алгоритма пользователя A300, включая два параллельно выполняемых процесса:

- действия пользователя типа:
запуск программы → наблюдение ее работы → останов программы;
- исполнение общего алгоритма A200 управлением сценарием.

Кроме того, представляет большой концептуальный и теоретический интерес проработка высокоуровневой интерпретации указанного низкоуровневого механизма прерывания программы (кнопкой stop):

понятной и полезной массовому программисту и, главное, пользователю.

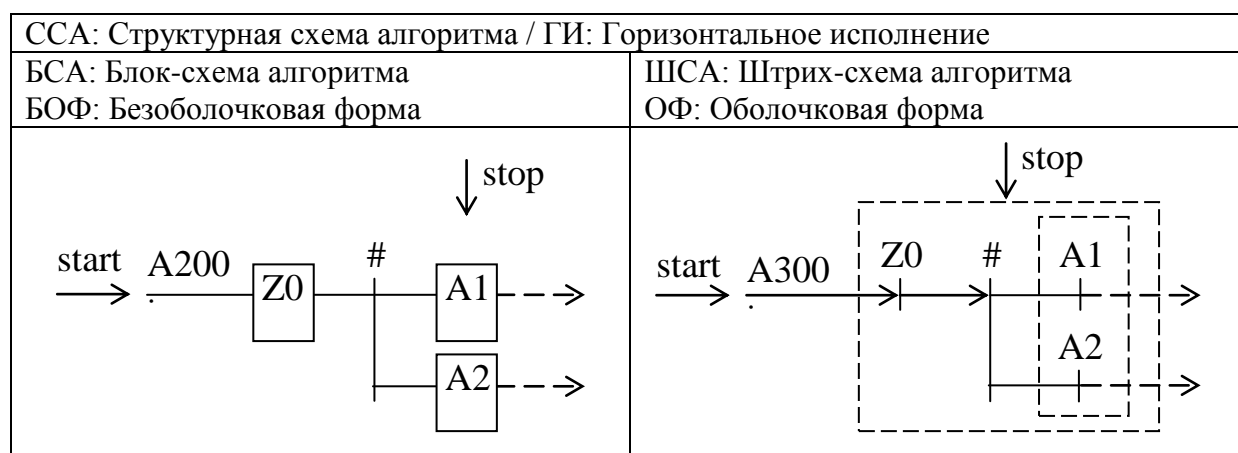


Рис. 1.6. Отражение сигналов пуска и останова

Обобщенный алгоритм (надалгоритм) управления системой моделирования

1.4 Конкретизация алгоритма по программным скриптам

Уточнение постановки алгоритмической задачи

Дано:

1) Общий (обобщенный) алгоритм сценария A100 программы и общий (обобщенный) алгоритм A200 управления сценарием.

2) Скриншоты окон построения скриптов активных объектов (спрайтов) программы (Рис. 1.7) – бес скриншота скрипта формирования сцены:

исходные коды программной реализации составляющих операторов A1, A2 общего алгоритма A100 в среде моделирования Scratch, реализованные в парадигме событийно-ориентированного программирования.

Надо:

1) Выполнить анализ скриптов (визуальных исходных кодов сценариев) объектов модели.

2) Выполнить конкретизацию алгоритма, реализуемого программой – применительно к принятой парадигме и языку программирования.

Скрипт объекта swing: качели 1 Программная реализация оператора частного алгоритма A1	Скрипт объекта seesaw: качели 2 Программная реализация оператора частного алгоритма A2
	

Рис. 1.7. Скрипты (исходные коды) моделируемых объектов

Диаграммы Насси-Шнейдермана

В окнах скриптов используется удобная в применении высокоавтоматизированная система интерактивного набора блоков команд и типовых синтаксических конструкций, которая использует принцип [диаграмм Насси – Шнейдермана](#): графический способ представления структурированных программ.

В среде Scratch нет вывода исходного кода (в окно консоли системы или на печать). Поэтому для анализа скриптов далее используется табличная имитация диаграмм Насси – Шнейдермана используемого типа (Рис. 1.8, Рис. 1.9):

Скрипты в данном случае – это специальная языковая надстройка (язык визуального программирования Scratch) над базовым языком объектно-ориентированного программирования (типа SmallTalk) для управления графическими объектами типа спрайтов в специальной графической среде программирования:

обеспечивается удобная и развитая система программирования на специализированном языке управления графикой, не требующем высокой квалификации программиста.

когда щелкнут по флагу (по кнопке пуск с флагом)	W0(f): Оператор ожидания события Пуск (f) Кнопка Пуск имитирует отмашку флагом (f: flag)
всегда	L: Оператор бесконечного цикла типа: While true
следующий костюм	R: Смена (reset) формы спрайта: следующий по замкнутому линейному порядку
ждать 1 сек	D(1): Задержка на 1 сек (delay)

Рис. 1.8. Скрипт объекта swing: качели 1

когда щелкнут по флагу	W0(f): Оператор ожидания события Пуск
всегда	L: Оператор бесконечного цикла типа: While true
перейти к костюму costume1	S(1): Установка (set) формы спрайта: costume1
ждать 1 сек	D(1): Задержка на 1 сек (delay)
перейти к костюму costume2	S(2): Установка (set) формы спрайта: costume2
ждать 1 сек	D(1): Задержка на 1 сек (delay)

Рис. 1.9. Скрипт объекта seesaw: качели 2

Уточнение структурной схемы общего алгоритма

Далее представлена фактическая блок-схема событийно-ориентированной реализации программы, эквивалентная (по управлению) исходной схеме.

На схеме (Рис. 1.10) показана эквивалентная система нелинейных переходов от неявного оператора A0 к операторам A1, A2 с индикацией обнаружения **события** Пуск (клик по кнопке Пуск), условно интерпретируемого как отмашка флага (f: flag = 1).

ССА: Структурная схема алгоритма / ГИ: Горизонтальное исполнение	
Исходная БСА: Блок-схема алгоритма	Эквивалентная БСА: Блок-схема алгоритма

Рис. 1.10. Нелинейные связи передачи управления по событию (f)

Уточнение структурной формулы общего алгоритма

Далее принимаются предварительные условные обозначения:

$A0(\uparrow f) = A_0(\uparrow^f)$ – неявный (скрытый) начальный оператор контроля и обработки события f с исходящей (выходной) поперечной стрелкой $\uparrow f = \uparrow^f$ нелинейной выходной передачи управления на последующие операторы:

этот оператор реализуется автоматически и в скриптах не отображается;

$A1(\downarrow f) = A_1(\downarrow^f)$, $A2(\downarrow f) = A_2(\downarrow^f)$ – операторы двух объектов сценария:

они реализуют нелинейную входную передачу управления $\downarrow f = \downarrow^f$ по входящей (входной) поперечной стрелке.

Соответственно этому уточняется структурная формула общего алгоритма управления сценарием:

$$A200 = A0 \rightarrow \#(A1, A2) = A0(\uparrow f) \mid (A1(\downarrow f), A2(\downarrow f)) = A_0(\uparrow^f)(A_1(\downarrow^f), A_2(\downarrow^f))$$

Возможна детализация операторов объектов сценария:

$$A1 = A1(\downarrow f) = W(f) \rightarrow A1' = W(f)A1';$$

$$A2 = A1(\downarrow f) = W(f) \rightarrow A2' = W(f)A2';$$

$$\text{где } A1' = L(R \rightarrow D(1)) = L(R - D(1)) = L(RD^1);$$

$$A2' = L(S(1) \rightarrow D(1) \rightarrow S(2) \rightarrow D(1)) = L(S(1) - D(1) - S(2) - D(1)) = L(S^1 D^1 S^2 D^1).$$

$W(f)$ – оператор ожидания (wait) события f ;

L – оператор свертки записи (потенциально бесконечной) однородной линейной последовательности операторов.

Знак секвенции ($\rightarrow = -> = -$) может опускаться: неявная секвенция.

Соответственно этому:

$$A1 = W(f)A1' = W(f)L(RD^1);$$

$$A2 = W(f)A2' = W(f)L(S^1 D^1 S^2 D^1).$$

Анализ полученных структурных формул

1.5 Текущие итоги

Выявляется общий алгоритм организации работы модельного сценария – функционирования двух периодических процессов.

Используется широко распространенный в параллельном программировании способ:

объекты запускаются на параллельное во времени исполнение процессов в бесконечном цикле – независимые (как в данном случае) или взаимодействующие между собой процессы (рассматриваются далее).

Используется (потенциально) бесконечное заикливание выполнения операторов, характерное для параллельного программирования, но не характерное для последовательного программирования.

Целесообразно введение читабельного псевдокода, как упрощенного исходного кода скриптов (согласованного со структурными формулами и схемами).

Получена обширная информация к размышлению во многих разных отношениях, требующая специального анализа и обсуждения.

В частности:

1) Насколько полезна и необходима подобная информация: учителям, преподавателям и студентам, использующим язык Scratch.

2) Насколько полезна такая информация школьникам разных возрастов: старшим, средним и младшим школьникам.

На сайте **paralg1000.narod.ru** (или **paralg.narod.ru**) – сейчас **paralg.ucoz.com** предусматривается представить подробный полиморфный анализ задачи в отношении многих согласованных между собой знаковых форм представления алгоритмов (с использованием разных псевдокодов алгоритмов и т.п.).

2 ПРИМЕР 2. ПРОСТАЯ ИГРОВАЯ АНИМАЦИЯ "FISHCHOMP"

2.1 Исходные данные

Объект алгоритмического анализа

Ниже (Рис. 2.1) приводится окно сцены среды программирования Scratch с загруженной более сложной программой **FishChomp**.

"Биологическая" модельная задача FishChomp (Глотание рыбок)	Содержание сцены программы
	На сцене представлена подводная среда и 4 действующих объекта (спрайта): 1) 3 маленькие золотые рыбки: goldfish – имеют красное тело; 2) 1 большая голодная рыба (хищник): hungry fish – имеет черную окантовку открытого рта. Красные и черные поля – сигнальные элементы взаимодействия.

Рис. 2.1. Окно сцены модельной среды Scratch с программой FishChomp

Содержание анимации. При запуске программы:

- маленькие рыбки (goldfish) начинают суетиться – периодически и часто случайным образом меняют направление движения;
- большая голодная рыба (hungry fish) начинает перемещаться с открытым ртом по направлению к курсору мыши;

При этом:

- если большая рыба касается черными губами красной маленькой рыбки, то она дважды закрывает рот и производит звук типа "чомп" (чавк) – имитируется глотание маленькой рыбки;
- маленькая рыбка исчезает, и через некоторое время в случайном месте появляется новая рыбка (это один и тот же видимый и невидимый спрайт).

Общий анализ работы программы

Далее (Рис. 2.2) приводятся формы (костюмы) спрайтов маленьких рыбок и большой рыбы. По ходу выполнения сценария объекты (спрайты) меняют направление движения и ориентацию:

- с поворотами в пространстве (маленькие рыбки);
- с зеркальным отражением спрайта (большая рыба) – при изменении положения курсора мыши относительно спрайта.

Постановка алгоритмической задачи

Аналогично предыдущему примеру принимаются следующие исходные условия.

Дано:

- 1) Программа FishChomp как объект алгоритмического анализа:
с наличием скриптов (данных и подпрограмм) действующих в программе объектов.
- 2) Краткое, приведенное выше, описание ее работы.

Надо. Как и ранее, представить поэтапное описание алгоритма, реализуемого программой FishChomp, включая этапы анализа:

- 1) Обобщенное описание *алгоритма сценария*.
 - 2) Обобщенное описание *алгоритма управления сценарием*.
 - 3) *Конкретизация алгоритма* по программным скриптам объектов.
- Далее эти вопросы излагаются очень кратко.

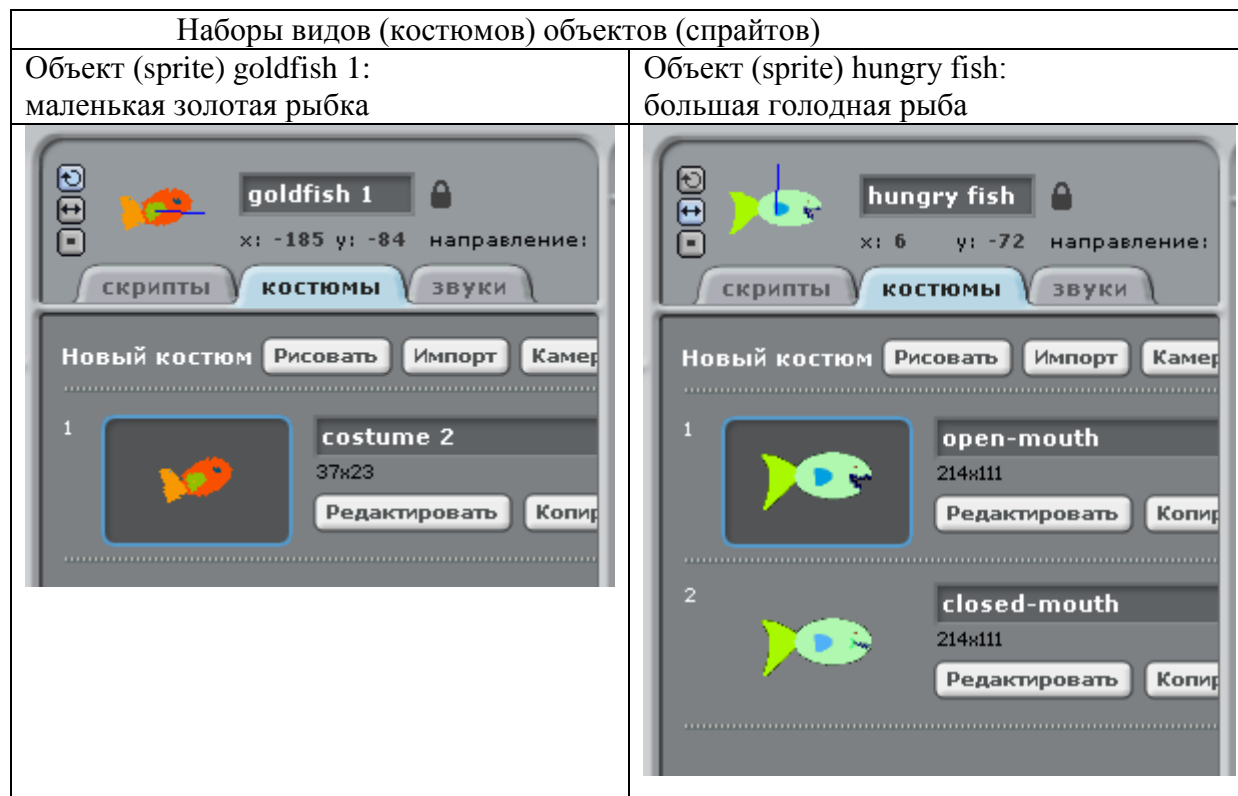


Рис. 2.2. Комплекты форм (костюмов) модельных объектов (спрайтов)

2.2 Обобщенное описание алгоритма управления сценарием

Исходные литерные обозначения

Принимаются (Табл. 2.1) условные обозначения общего оператора алгоритма управления сценарием и составляющих его операторов алгоритмом участвующих объектов.

Структура данного алгоритма A200 подобна аналогичному алгоритму предыдущего примера, но имеет дополнительные особенности:

- общий алгоритм A100 сценария имеет 4 составляющих алгоритма A_i ($i = 1..4$), которые определяют (потенциально) бесконечные периодические процессы: действий маленьких рыбок A_i , $i = 1..3$ и большой рыбы A4;
- модельные процессы маленьких рыбок A1..A3 взаимодействуют с модельным процессом большой рыбы A4, что условно представлено дополнительными горизонтальными связями взаимодействия: они реагируют на событие касания красного и черного цвета (касание красного тела маленькой рыбки и черного рта большой рыбки).

Табл. 2.1. Исходные обозначения

ЛТА: Литерный текст алгоритма	// ЛЗФ: Литерная знаковая форма
УЛО: Условные литерные обозначения	
ООА: Общее обозначение (оператора) алгоритма сценария	
A100: Поймать (съесть) рыбку: Fish Chomp:	
ООА: Общее обозначение алгоритма управления сценарием	
A200: управление: общее управление сценарием	
КСА: Комплект составляющих (операторов) алгоритмов	
A0: ПС: Подготовка сценария	
A_i : МРi: Плавание маленькой рыбки (goldfish i), $i = 1..3$:	
A4: БР4: Плавание большой рыбы (hungry fish)	

Структурная схема общего алгоритма управления сценарием

Далее (Рис. 2.3) сразу представлена блок-схема общего алгоритма A200 управлением сценарием, включающая в себя общий алгоритм сценария A100.

Структурная формула алгоритма управления сценарием

Далее представлено литерное описание алгоритма.

1) Общий алгоритм сценария – комплект алгоритмов объектов:

$$A100 = (A1, A2, A3, A4) = \\ = (A1(\uparrow^1), A2(\uparrow^2), A3(\uparrow^3), A4(\uparrow^{1,2,3})) = (A1(\uparrow^1), A2(\uparrow^2), A3(\uparrow^3), A4(\uparrow^{1,2,3})).$$

2) Общий алгоритм управления сценарием:

$$A200 = (A0 \rightarrow \# A100) = (A0 \rightarrow \# (A1, A2, A3, A4)) = \\ = (A0 \rightarrow \# (A1(\uparrow^1), A2(\uparrow^2), A3(\uparrow^3), A4(\uparrow^{1,2,3}))) = \\ = (A0 \rightarrow \# ((A1(\uparrow^1), A2(\uparrow^2), A3(\uparrow^3), A4(\uparrow^{1,2,3}))).$$

Знак типа \uparrow^i условно обозначает наличие дополнительных связей взаимного влияния операторов (нелинейная передача управления).

Представляет интерес конкретизация и детализация алгоритма по скрипам объектов по аналогии с предшествующим примером.

ГТА: Графический текст алгоритма // ГЗФ: Графическая знаковая форма
 ССА: Структурная схема алгоритма / ВГ: Векторная графика
 БСА: Блок-схема алгоритма (ССА = БСА) / ВИ: Вертикальное исполнение

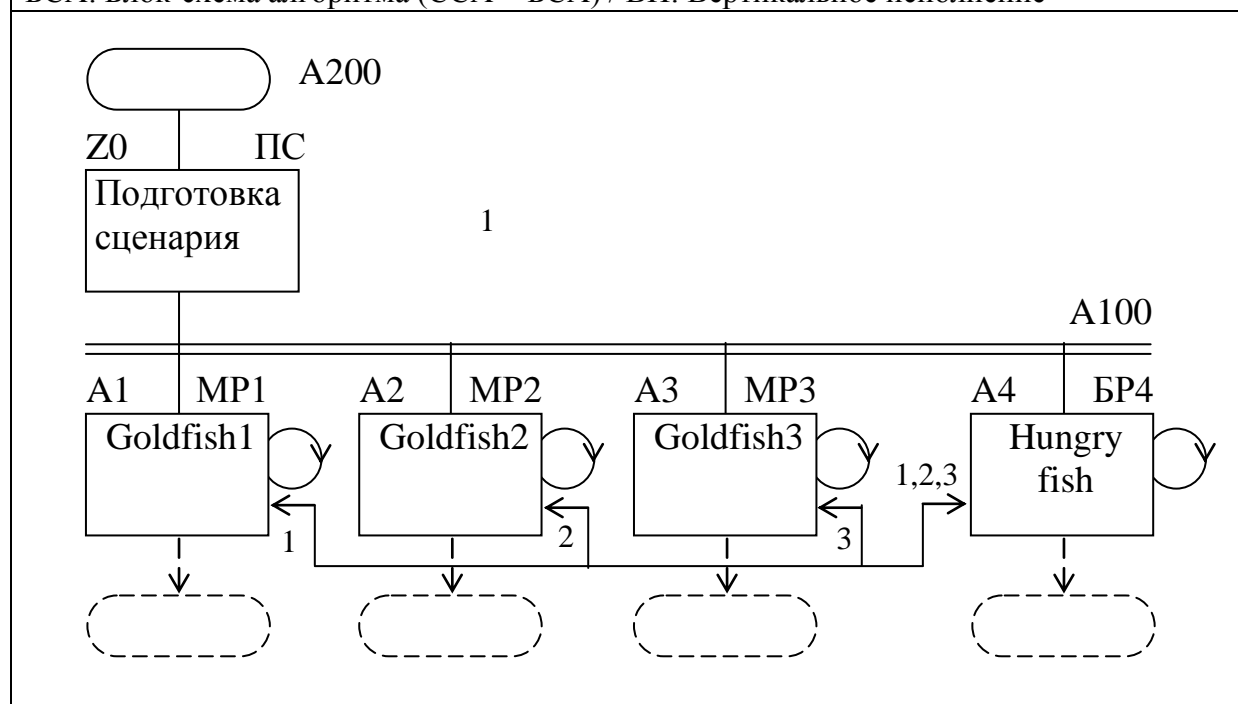


Рис. 2.3. Общая схема алгоритма управления сценарием

ЗАКЛЮЧЕНИЕ

Выполненный предварительный анализ простых (детских) примеров с некоторым множеством независимых и взаимодействующих объектов (спрайтов).

Далее приводятся предварительные выводы:

1) Выявляется применение здесь (в приведенных примерах) **типовых исходных структур** модельной организации их **параллельной во времени работы** объектов:

- или с полностью **независимыми** параллельными процессами;
- или с наличием **взаимодействия** процессов – с его реализацией средствами некоторого варианта событийного программирования.

2) Простыми и наглядными средствами реализуется **первый (и главный) принцип** в направлении **объектно-ориентированного программирования**:

- все действующие графические объекты имеют свои скрипты, которые содержат общие и индивидуальные переменные и представляют собой одну или несколько подпрограмм, что примерно соответствует свойствам и методам объектов;
- классов таких объектов нет – нет управления тиражированием, обслуживанием и ликвидацией объектов, нет наследования, полиморфизма, переопределения и т.п.;
- тем не менее, **реализуется главное свойство – автономное поведение объектов**;
- возможно тиражирование объектов методом копирования – с переименованием объектов и, может быть, некоторых или всех переменных.

3) Представляет интерес **интерактивная система блочного набора** текстов скриптов, включая следующие ее особенности:

а) Система графических блоков литерных текстов скриптов, реализующих условные синтаксические обозначения **диаграмм Насси – Шнейдермана**.

б) **Автоматизация** комплекса вспомогательных графических синтаксических функций:

- адаптация (автоматическая подстройка) растворов цикловых и других блоков по длине (высоте) вложенных блоков текста скриптов;
- автоматический графический синтаксический контроль - обеспечивается сборка или не-сборка сопрягаемых блоков по соответствию или несоответствию фигур графических форм или их элементов.

4) Первый опыт работы со средой моделирования Scratch показал:

- **многократное сокращение объема исходного кода и трудоемкости** программирования по сравнению с применением обычных высокоуровневых языков программирования с графическими возможностями;
- это связано с наличием простой специализированной графической среды программирования.

5) Есть определенные функциональные ограничения (пока не очень ясные по памяти – необходим анализ).

6) Проблемой (в принципе разрешимой) является отсутствие возможности вывода на печать (на экран монитора или на бумагу) текста исходного кода программ.

Автор статьи (по ходу первичного изучения это графического языка программирования Scratch) довольно быстро "набросал" порядка десяти простых модельных технических программ, в частности, с параллелизмом, по аналогии с ранее реализованными программами на других языках программирования:

для сравнительного анализа и демонстраций на занятиях по программной реализации параллельных алгоритмов в разных парадигмах (предполагается их анализ на сайте).

ЛИТЕРАТУРА

Статьи [4,5] автора по параллельной тематике доступны на сайте: paralg.ucoz.com: Параллельные алгоритмы и логика.

1. Википедия. Статья: [Scratch \(язык программирования\)](#).
2. <http://scratch.mit.edu/> Официальный многоязычный сайт международного проекта: Scratch (есть русскоязычное отделение):
3. <http://scratch.ucoz.net/> Инициативный сайт: Scratch (русскоязычный):
4. Житников А.П., Житников В.П., Шерыхалина Н.М. [Программно-методический комплекс "Параллельные алгоритмы и программы"](#) / Материалы научно-технического совещания "Высокопроизводительные вычислительные ресурсы России: состояние и перспективы развития". – Уфа: УГАТУ, 2003. С. 151 – 161.
5. Житников А.П. [Базовая структурная семантика параллельных алгоритмов](#). // Принятие решений в условиях неопределенности. Межвуз. науч. сборник. – Уфа: УГАТУ, 2003. – С. 121 – 130.

