Санкт-Петербургский государственный институт точной механики и оптики
(технический университет)

Кафедра «Компьютерные технологии»

**В.П. Пенев, В.В. Степаненков,
Е.А. Сучкоусов, А.А. Шалыто**

Компьютерная игра

«Automatic Bomber»

Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург

2003

СОДЕРЖАНИЕ

1. Предыстория

История создания этой игры уходит корнями в недалекое прошлое, когда компания Microsoft еще не захватила весь мир своей операционной системой Windows, и все программисты на персональных компьютерах писали игры под DOS. Люди очень любили маленькие ДОСовские игрушки, которые умещались на одной дискете и поэтому свободно кочевали с компьютера на компьютер. Одной из таких игр стала игра “Mr. Boom”, написанная неизвестными авторами, связаться с которыми нам так и не удалось. Мы долго играли в эту интерсную, быструю, сетевую игру пока не наступила эра Windows NT. К сожалению, наша любимая игра не хотела работать под этой операционной системой. И мы решили написать точно такую же игру, но уже под Windows NT, используя графику прототипа.

2. Введение

Данный проект разработан в рамках курса «Теория автоматов в программировании». Суть проекта заключается в создании аналога игры «Mr. Boom» на основе объектно-ориентированного программирования с явным выделением состояний. Этот подход предложен в работе Шалыто А.А., Туккель Н.И. Система управления танком для игры Robocode. Объектно-ориентированное программирование с явным выделением состояний. (http://is.ifmo.ru).

3. Правила игры

В игру могут играть одновременно от двух до восьми человек - игроков. Каждый игрок видит на экране поле, на котором могут находиться:

- персонажи (маленькие человечки);
- монстры (если игроков меньше восьми);
- бомбы.

Персонажи и монстры могут двигаться в различных направлениях. Каждый игрок управляет своим персонажем с помощью клавиатуры. Монстры управляются компьютером.

У каждого персонажа имеются бомбы, которыми он может взрывать противников. Для этого он должен установить бомбу в какой-нибудь клетке поля. Через три секунды она взорвется. Тот игрок, чей персонаж остался на поле один, объявляется победителем. В начале игры персонаж может поставить не более одной бомбы – пока не взорвется поставленная бомба, следующую бомбу он поставить не может.

Монстры предназначены для поддержания интереса в игре, если персонажей меньше восьми. При наличии восьми персонажей монстры на игровом поле не появляются. Цель монстров – убить персонажей. Прикосновения монстров для персонажей смертельно опасны, но монстра можно уничтожить, взорвав рядом с ним бомбу. Каждый монстр имеет от одной до трех жизней. Количество жизней мостров априори не известно. Поэтому некоторых монстров для полного уничтожения требуется взорвать более одного раза. Если монстр еще не уничтожен, то он при ранении мигает несколько секунд.

На поле имеются стены, которые мешают персонажам двигаться. Стены бывают двух типов – одни могут взрываться, а другие – не могут.

Игра имеет шесть различных игровых полей. Переход с поля на поле осуществляется автоматически после победы одного из персонажей на предыдущем поле. Все персонажи начинают играть на новом поле в равных условиях – с одной бомбой, обладающей малой силой взрыва (в одну клетку).

4. Управление

Для управления персонажем игроку требуется пять клавиш: для управления движением – четыре клавиши и одна - для установки бомбы. На данный момент установлены следующие клавиши управления.

Игрок 1:

- <S> - движение влево;
- <D> - движение вниз;
- <F> - движение вправо;
- <E> - движение вверх;
- <Q> - установка бомбы.

Игрок 2:

- <стрелка влево> - движение влево;

- <стрелка вправо> - движение вправо;
- <стрелка вниз> - движение вниз;
- <стрелка вверх> - движение вверх;
- <O> - установка бомбы.

5. Версии игры

В настоящей работе описывается версия 1.0 игры. Реализована возможность игры вдвоем на одном компьютере. Вместо каждого из остальных шести персонажей появляются монстры. Реализован автоматический переход между уровнями после гибели всех монстров. При смерти одного из персонажей игра заканчивается – происходит выход в меню. Всего в игре реализовано шесть различных уровней.

6. Постановка задачи

Цель настоящей работы состоит в создании игры, отвечающей описанным выше правилам, на основе объектно-ориентированного программирования с явным выделением состояний.

Внешний вид первого уровня игры приведен на рис. 1.

Рис. 1. Фрагмент игры

7. Структура программы

Код программы написан на языке C++. Классы в программе взаимодействуют с применением механизма передачи событий. Этот механизм реализован с использованием класса «Событие» и класса «Очередь событий». Список всех событий приведен в Приложении 1. Каждый объект может посылать событие в очередь, «не беспокоясь» о том дойдет ли оно до адресата. Каждый класс, в котором необходимо обрабатывать события, содержит управляющий автомат, первым получающий события и осуществляющий их обработку. При необходимости управляющий автомат передает это событие другому классу. Кроме управляющих автоматов в классах могут содержаться один или несколько автоматов других типов, которые предназначены не для обработки событий, а для управления самим объектом, в котором они содержатся. При этом в каждом классе управляющий автомат вызывает другие автоматы.

8. Диаграмма классов

Диаграмма классов приведена на рис.2.

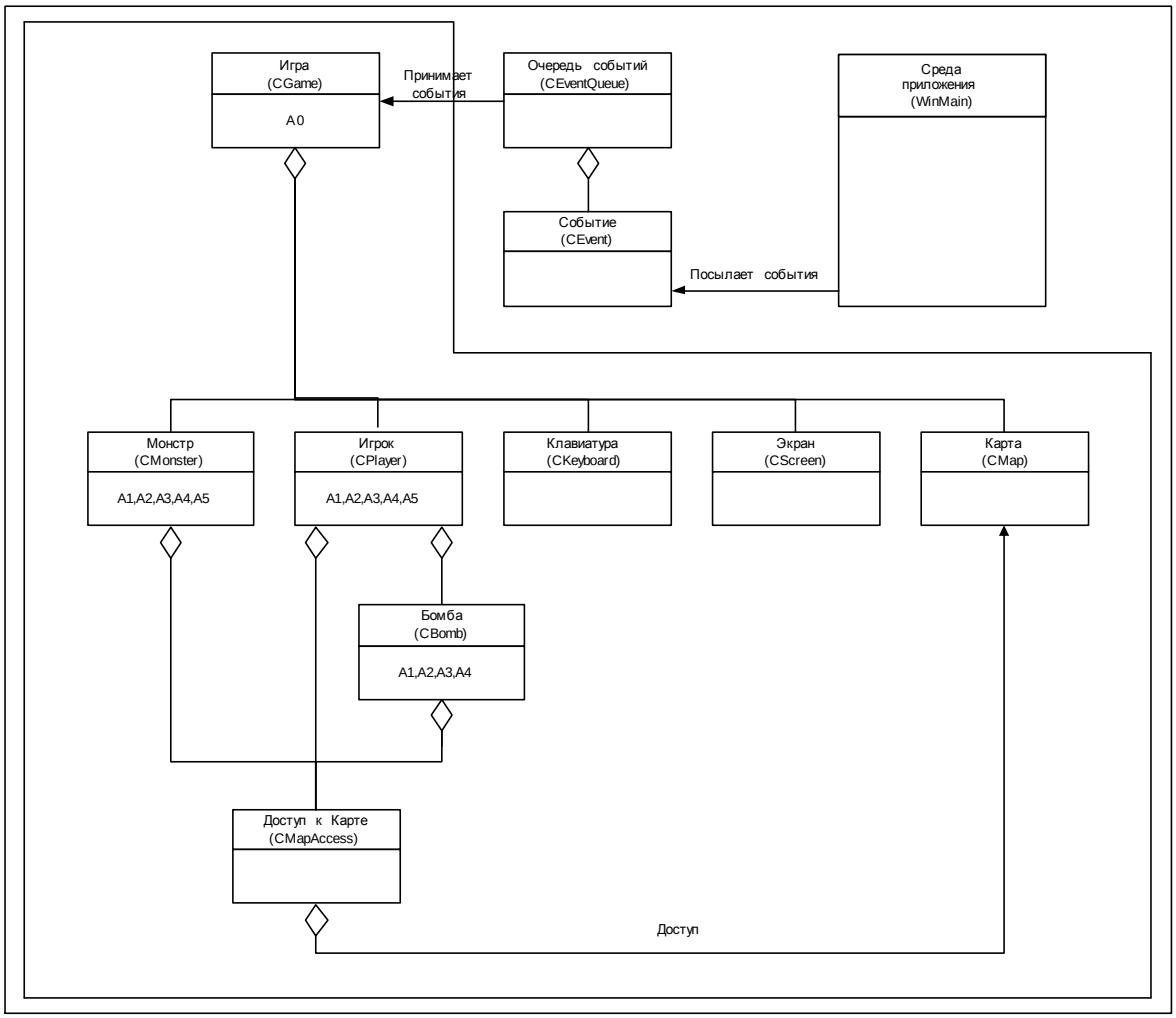

Рис. 2. Диаграмма классов

«Среда приложения» (главный цикл) генерирует определенные правилами игры события и записывает их в объект «Очередь событий». Перед тем, как передать управление в иерархию классов, приложение начинает разбор очереди событий. Объект «Событие» из очереди удаляется и передается в управляющий автомат A0 класса «Игра». Если это событие влияет только на класс «Игра», то после обработки оно уничтожается. В противном случае это событие передается по иерархии классов дальше. Все классы в иерархии могут формировать события и обмениваться ими. Иерархия классов выделена линией. Классы «Событие» и «Очередь событий» обеспечивают хранение событий до их обработки.

Отметим, что если события в программе в целом имеют только уникальные обозначения (номера), то автоматы, их входные переменные и выходные воздействия имеют уникальные обозначения только внутри класса.

В настоящей работе рассматриваются только автоматные классы.

9. Класс «Игра» (CGame)

9.1. Словесное описание

Класс «Игра» является головным в иерархии классов. Основной задачей этого класса является общее управление программой, передача событий другим классам в иерархии и взаимодействие со «Средой приложения». Он является системозависимым в отличие от других «автоматных» классов в иерархии.

9.2. Структурная схема класса

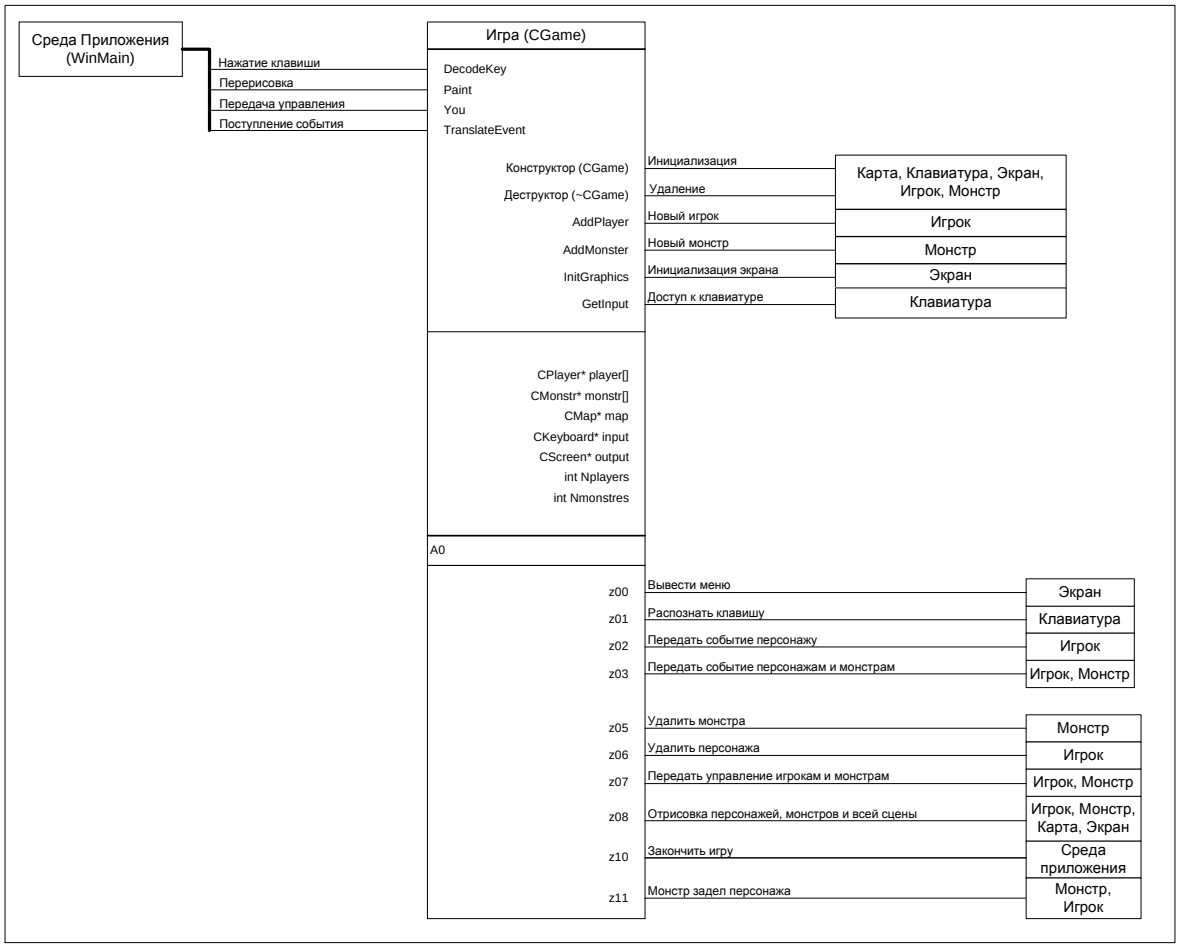

Рис. 3. Структурная схема класса «Игра»

9.3. Управляющий автомат (A0)

9.3.1. Словесное описание

Автомат обеспечивает переключение экранов в игре (с экрана «Меню» на экран «Игра» и обратно). В экране «Меню», куда автоматически попадает игрок после запуска игры, автомат управляет курсором меню, а также обеспечивает выполнение пунктов меню. Когда игрок активизирует пункт меню «начать игру», происходит переключение на экран «Игра». Начинается игра. После завершения игры происходит обратное переключение на экран «Меню».

9.3.2. Схема связей

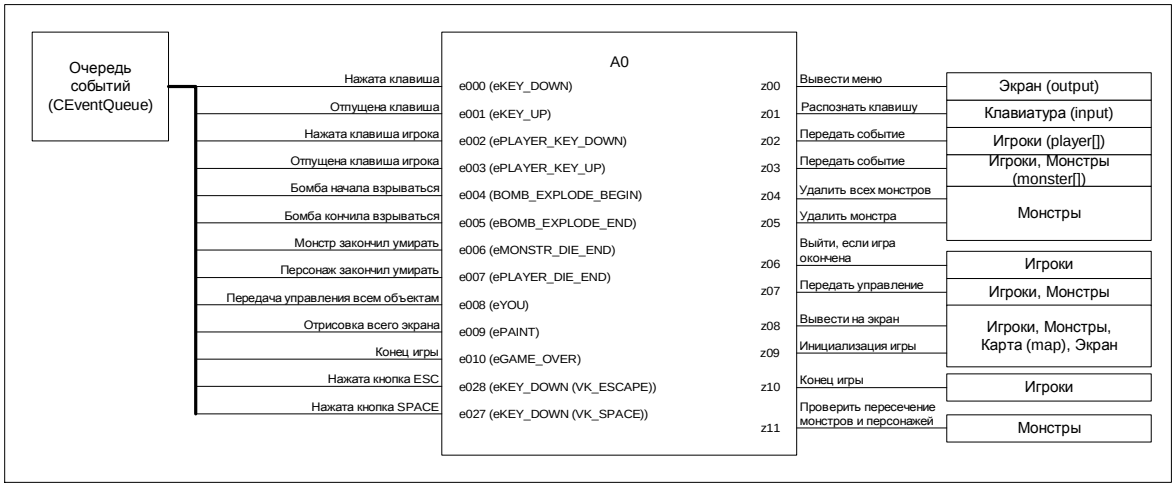

Рис. 4. Схема связей управляющего автомата (A0) класса «Игра»

9.3.3. Граф переходов

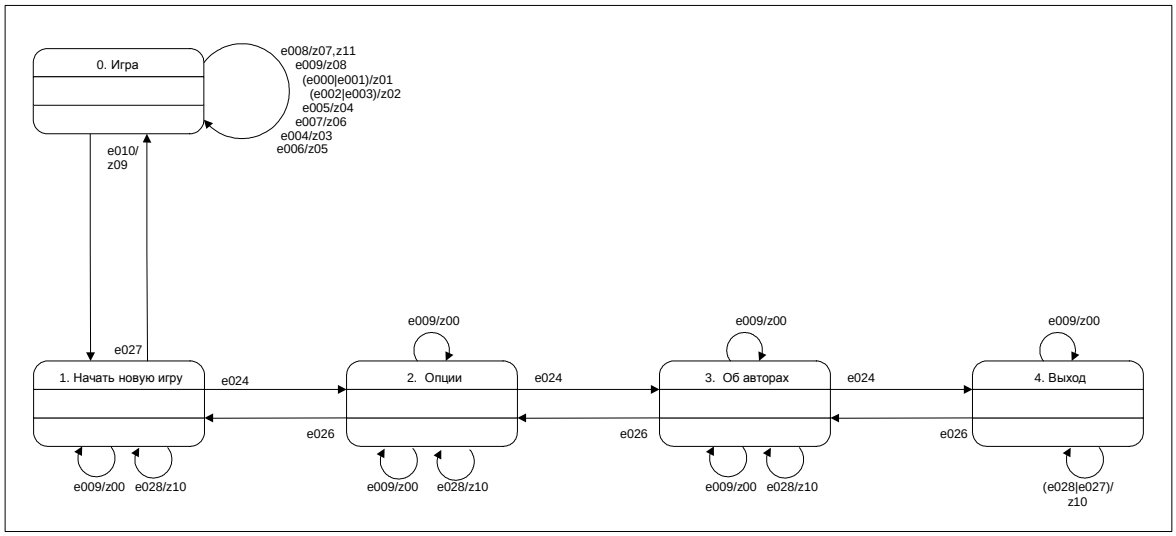

Рис. 5. Граф переходов управляющего автомата (A0) класса «Игра»

10. Класс «Игрок» (CPlayer)

10.1. Словесное описание

Класс «Игрок» управляет действиями персонажа в процессе игры. Он хранит всю информацию о персонаже, например выделенные для него клавиши управления, положение на экране и количество оставшихся жизней.

10.2. Структурная схема класса

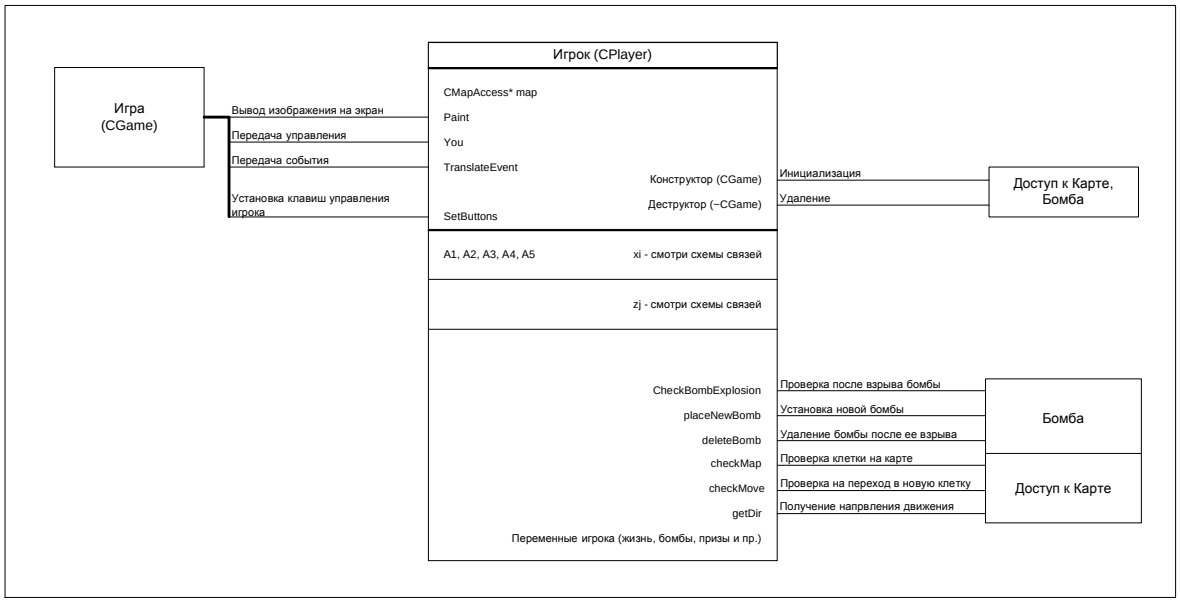

Рис. 6. Структурная схема класса «Игрок»

10.3. Схема взаимодействия автоматов

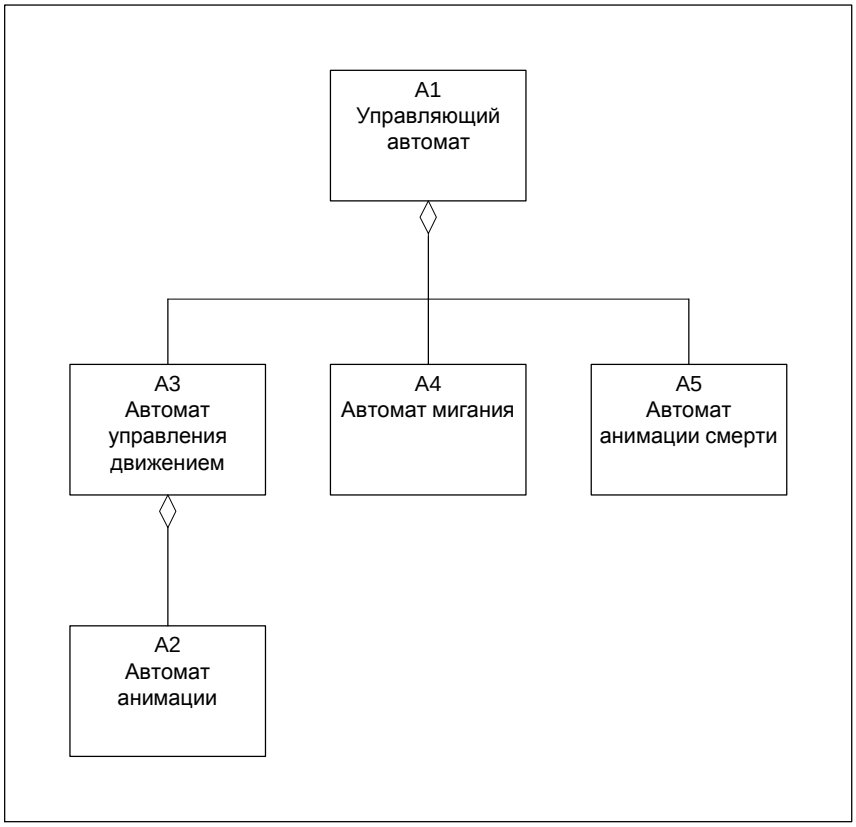

Рис. 7. Схема взаимодействия автоматов класса «Игрок»

10.4. Управляющий автомат (A1)

10.4.1. Словесное описание

Автомат является главным в классе «Игрок» и управляет всеми его действиями. Он обрабатывает все события, приходящие «Игроку», и вызывает другие автоматы этого класса.

10.4.2. Схема связей

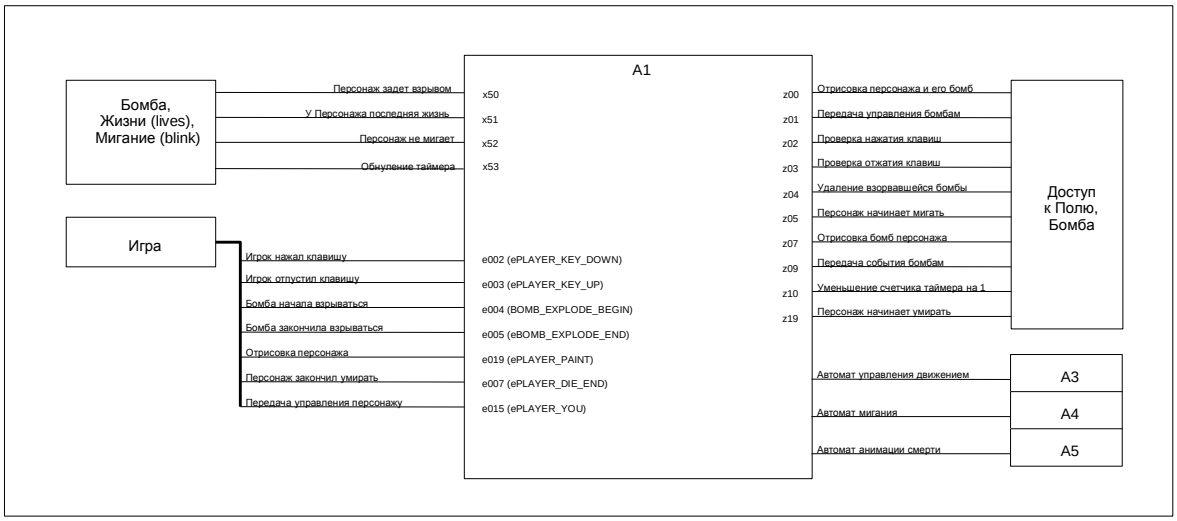

Рис. 8. Схема связей управляющего автомата (A1) класса «Игрок»

10.4.3. Граф переходов

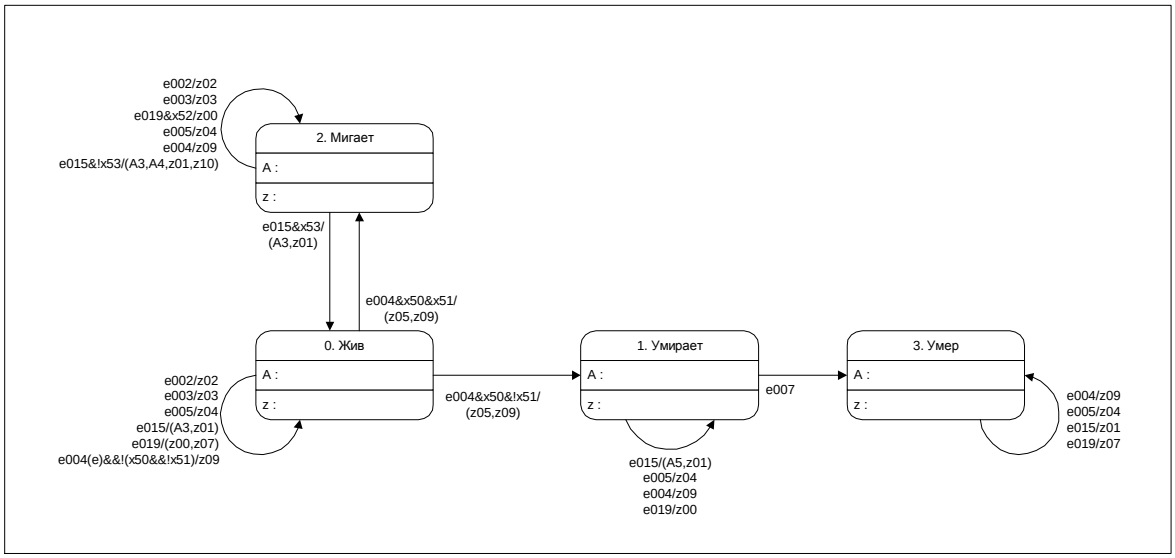

Рис. 9. Граф переходов управляющего автомата (A1) класса «Игрок»

10.5. Автомат анимации (A2)

10.5.1. Словесное описание

Автомат управляет сменой кадров при движении персонажа. Всего у персонажа три кадра. Их смена производится при равенстве значения «Счетчика анимации» нулю не последовательно, а по схеме: 0, 1, 0, 2. После смены кадра счетчику присваивается исходное значение, равное задержке кадра.

10.5.2. Схема связей

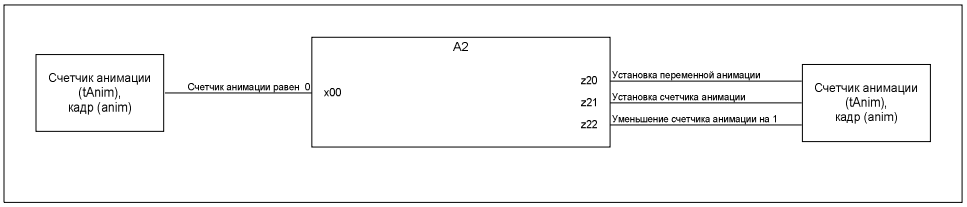

Рис. 10. Схема связей автомата анимации (A2) класса «Игрок»

10.5.3. Граф переходов

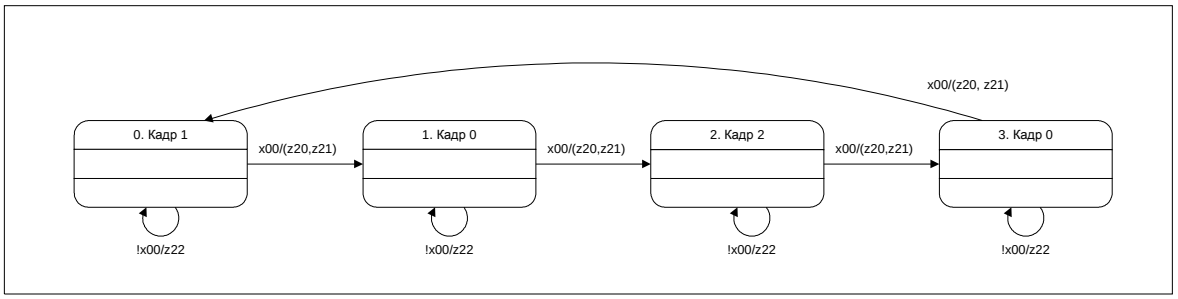

Рис. 11. Граф переходов автомата анимации (A2) класса «Игрок»

10.6. Автомат управления движением (A3)

10.6.1. Словесное описание

Автомат управляет персонажем при нажатии клавиш управления. Он проверяет столкновение со стенками и бомбами на поле.

10.6.2. Схема связей

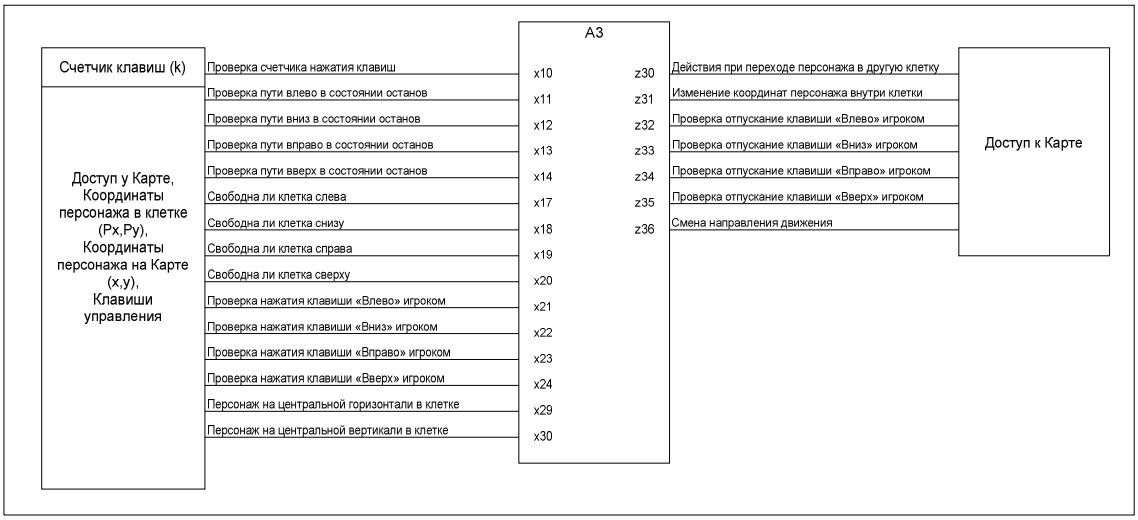

Рис. 12. Схема связей автомата управления движением (A3) класса «Игрок»

10.6.3. Граф переходов

На графе переходов для его обозримости дуги обозначены символами c01 - c24. Условия и действия на дугах приведены в таблице 1.

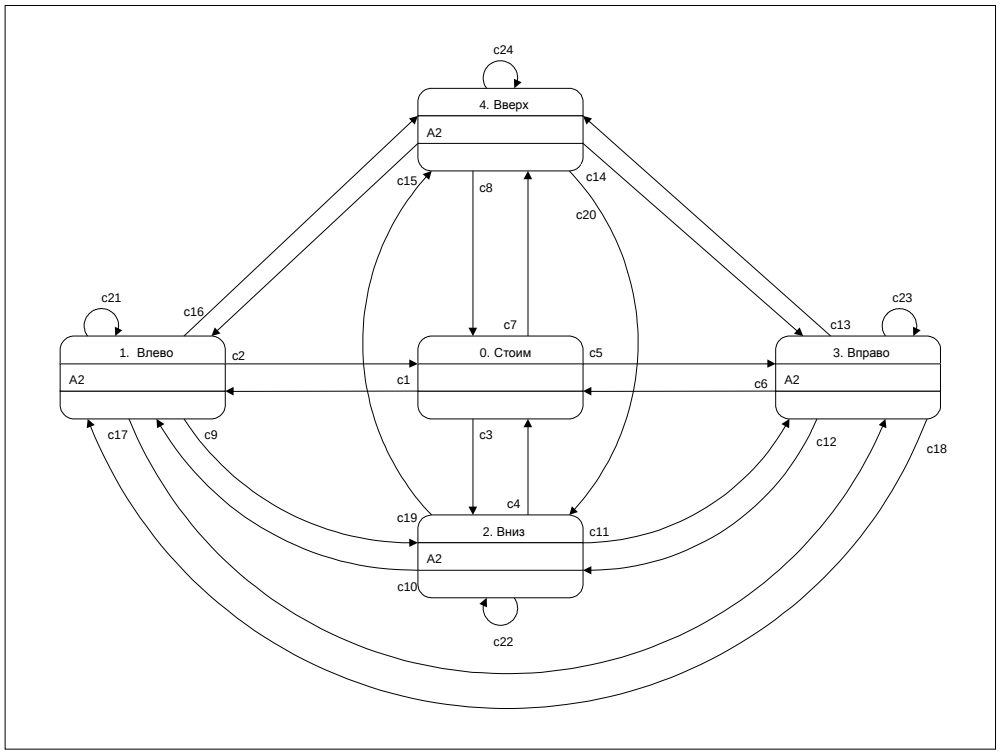

Рис. 13. Граф переходов автомата управления движением (A3) класса «Игрок»

Таблица 1. Условия и действия на дугах в автомате управления движением (А3) класса «Игрок»

Дуга	Условия	Действия
c1	x11	z31, z30
c2	(!x17 & x29)|x10	z36
c3	x12	z31, z30
c4	(!x18 & x30)|x10	z36
c5	x13	z31, z30
c6	(!x19 & x29)|x10	z36
c7	x14	z31, z30
c8	(!x20 & x30)|x10	z36
c9	x29 & x18 & x22	z31, z30, z32
c10	x30 & x17 & x21	z31, z30, z33
c11	x30 & x19 & x23	z31, z30, z33
c12	x29 & x18 & x22	z31, z30, z34
c13	x29 & x20 & x24	z31, z30, z34
c14	x30 & x19 & x23	z31, z30, z35
c15	x30 & x17 & x21	z31, z30, z35
c16	x29 & x20 & x24	z31, z30, z32
c17	x19 & x23	z31, z30, z32
c18	x17 & x21	z31, z30, z34
c19	x20 & x24	z31, z30, z33
c20	x18 & x22	z31, z30, z35
c21	1 (всегда)	z31, z30
c22	1	z31, z30
c23	1	z31, z30
c24	1	z31, z30

10.7. Автомат мигания (А4)

10.7.1. Словесное описание

Автомат управляет процессом мерцания, когда персонажа задело взрывом, но жизней у него больше, чем одна. Автомат последовательно переключает состояния персонажа.

10.7.2. Схема связей

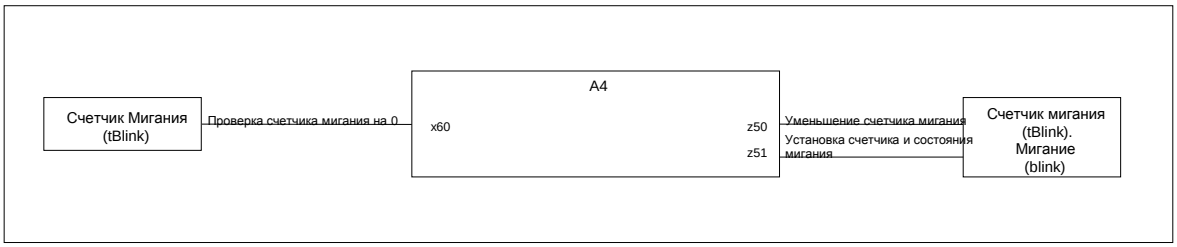

Рис. 14. Схема связей автомата мигания (A4) класса «Игрок»

10.7.3. Граф переходов

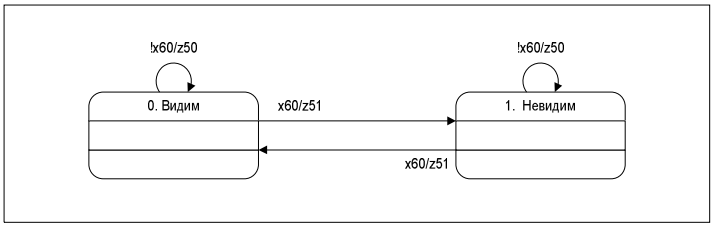

Рис. 15. Граф переходов автомата мигания (A4) класса «Игрок»

10.8. Автомат анимации смерти (A5)

10.8.1. Словесное описание

Автомат управляет сменой кадров, анимирующих смерть персонажа. Кадры меняются последовательно. В конце работы автомат посылает сообщение о смерти персонажа (выходное воздействие z62).

10.8.2. Схема связей

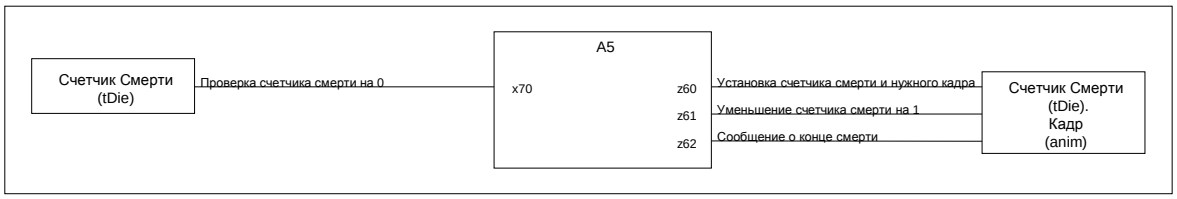

Рис. 16. Схема связей автомата анимации смерти (A5) класса «Игрок»

10.8.3. Граф переходов

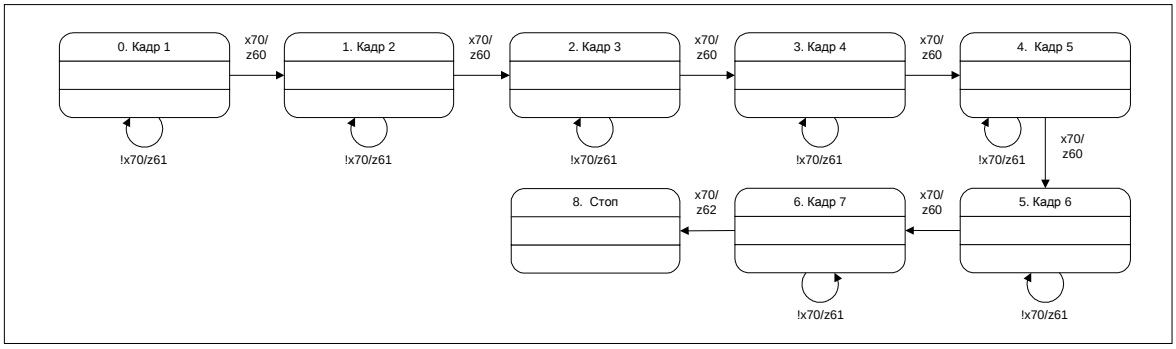

Рис. 17. Граф переходов автомата анимации смерти (A5) класса «Игрок»

11. Класс «Монстр» (CMonster)

11.1. Словесное описание

Класс «Монстр» управляет действиями всех монстров в процессе игры. Он хранит всю информацию о монстре. В процессе игры монстры могут двигаться, ранить персонажей и умирать. Для разных типов монстров применяются различные алгоритмы управления ими. В этой версии игры реализован один тип монстров.

11.2. Структурная схема класса

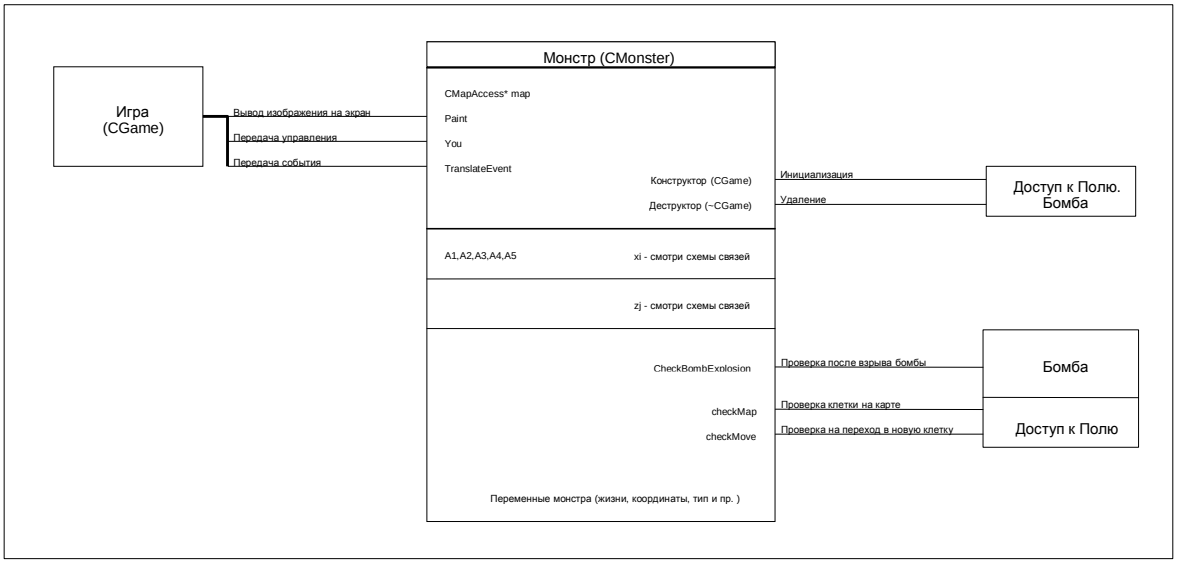

Рис. 18. Структурная схема класса «Монстр»

11.3. Схема взаимодействия автоматов

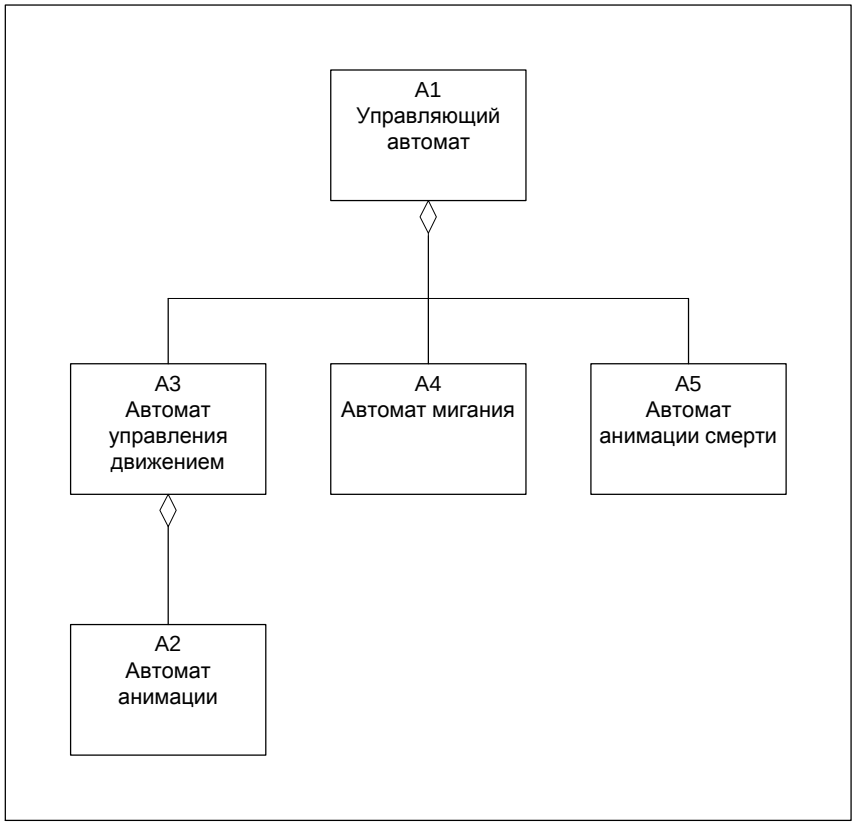

Рис. 19. Схема взаимодействия автоматов класса «Монстр»

11.4. Управляющий автомат (A1)

11.4.1. Словесное описание

Автомат A1 является главным в классе «Монстр». Он управляет всеми действиями монстра и обрабатывает все события, приходящие монстру от класса «Игра». Управляющий автомат также вызывает другие автоматы в классе «Монстр».

11.4.2. Схема связей

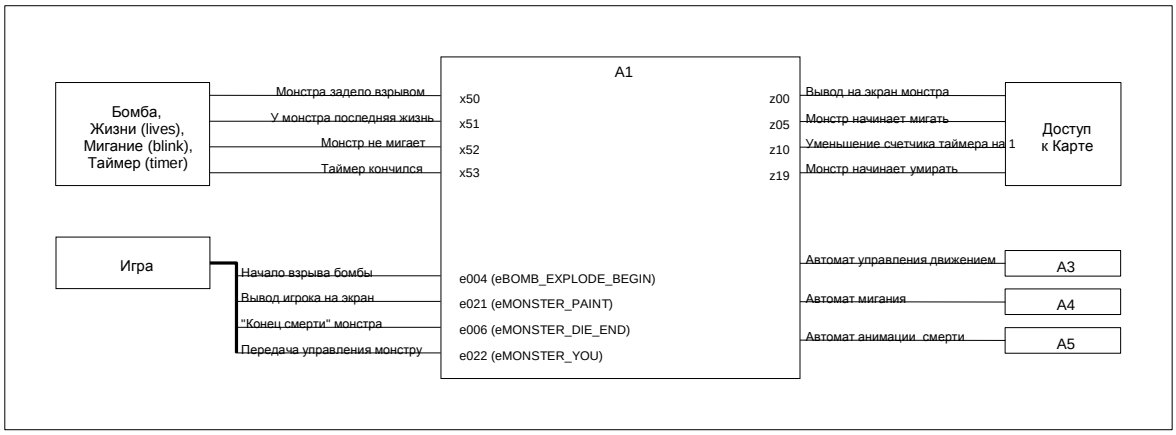

Рис. 20. Схема связей управляющего автомата (A1) класса «Монстр»

11.4.3. Граф переходов

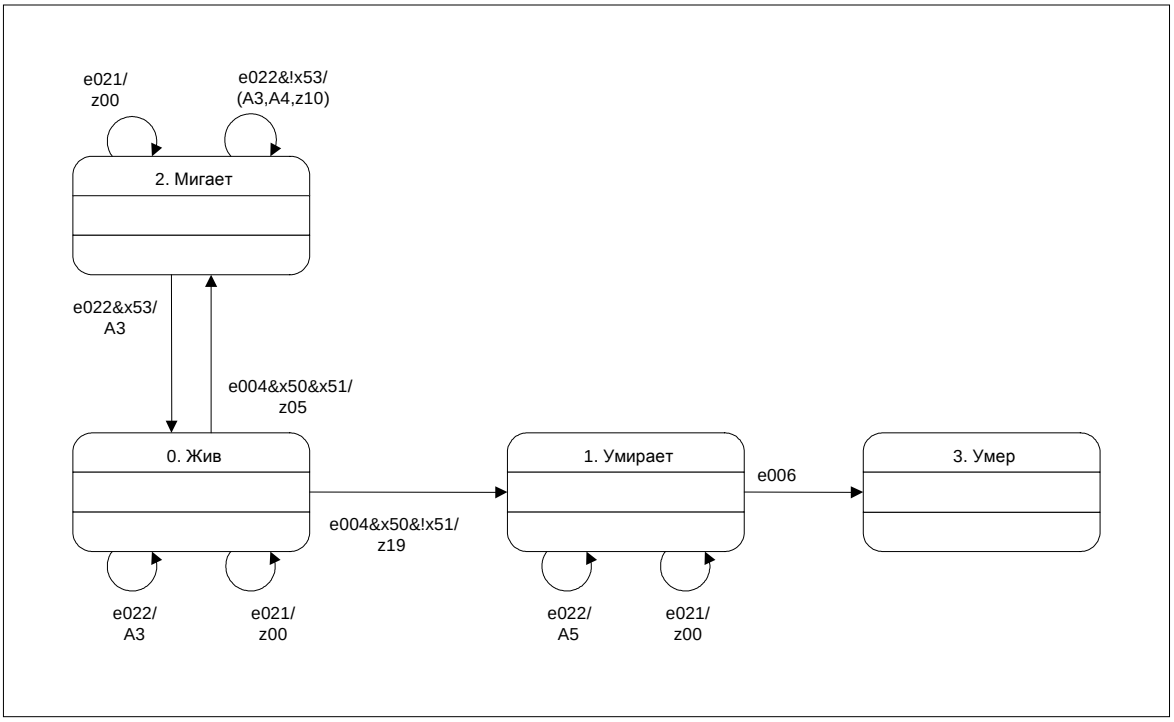

Рис. 21. Граф переходов управляющего автомата (A1) класса «Монстр»

11.5. Автомат анимации (A2)

11.5.1. Словесное описание

Автомат управляет сменой кадров при движении монстра. Монстр анимируется тремя кадрами по схеме: 0, 1, 0, 2.

11.5.2. Схема связей

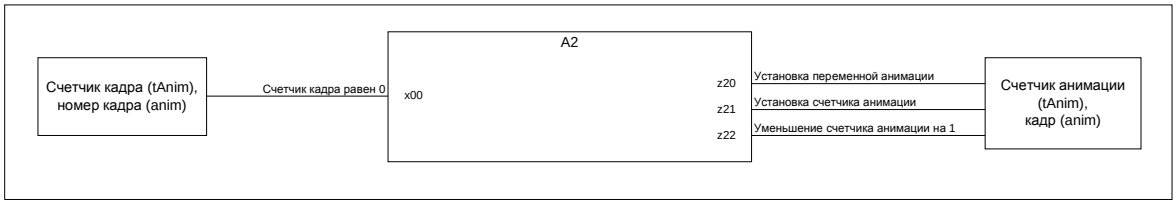

Рис. 22. Схема связей автомата анимации (A2) класса «Монстр»

11.5.3. Граф переходов

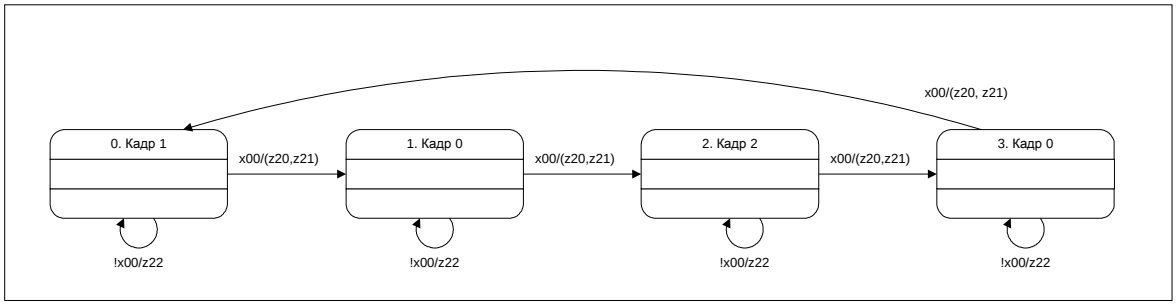

Рис. 23. Граф переходов автомата анимации (A2) класса «Монстр»

11.6. Автомат управления движением (A3)

11.6.1. Словесное описание

Автомат управляет движением монстра. Это интеллект монстра. В этой версии программы монстры очень тупые. Они в каждый момент времени случайным образом выбирают куда идти. В дальнейшем планируется усложнить логику. При этом монстры будут способны уклоняться от взрывов и стремиться убить персонажей.

11.6.2. Схема связей

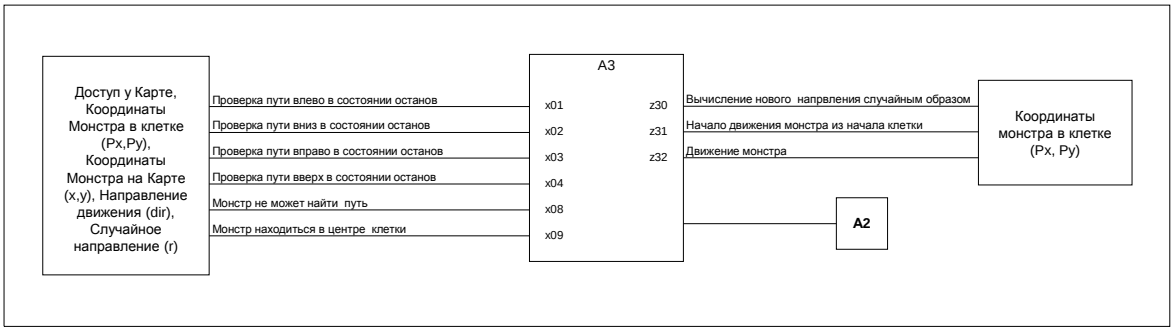

Рис. 24. Схема связей автомата управления движением (A3) класса «Монстр»

11.6.3. Граф переходов

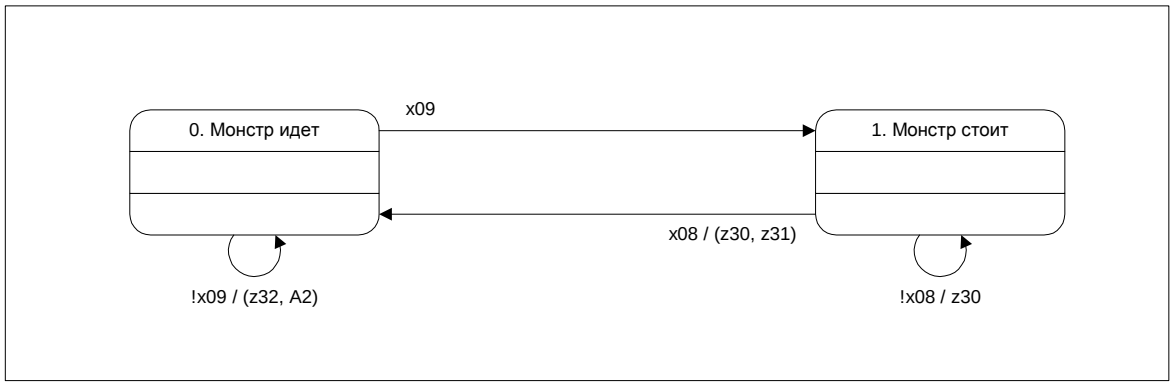

Рис. 25. Граф переходов автомата управления движением (A3) класса «Монстр»

11.7. Автомат мигания (A4)

11.7.1. Словесное описание

Автомат управляет процессом мерцания, когда монстра задело взрывом, но жизни у него еще остались. Автомат будет последовательно переключать состояния монстра.

11.7.2. Схема связей

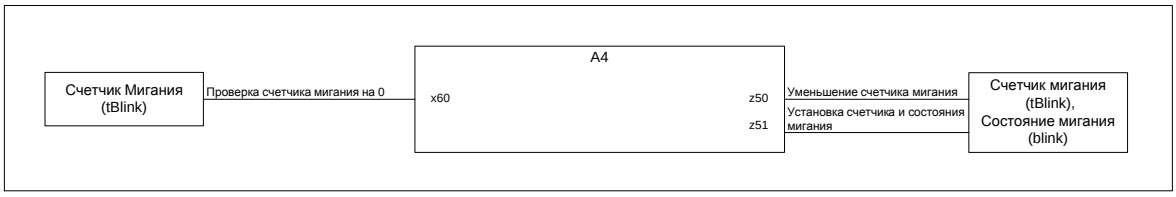

Рис. 26. Схема связей автомата мигания (A4) класса «Монстр»

11.7.3. Граф переходов

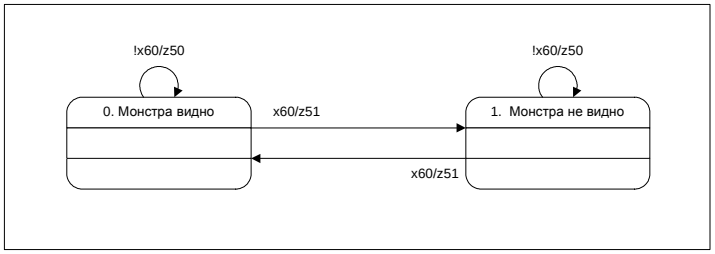

Рис. 27. Граф переходов автомата мигания (A4) класса «Монстр»

11.8. Автомат анимации смерти (A5)

11.8.1. Словесное описание

Автомат управляет сменой кадров при смерти монстра. Кадры меняются последовательно. В конце работы автомат посылает событие, сообщающее о смерти монстра (выходное воздействие z62).

11.8.2. Схема связей

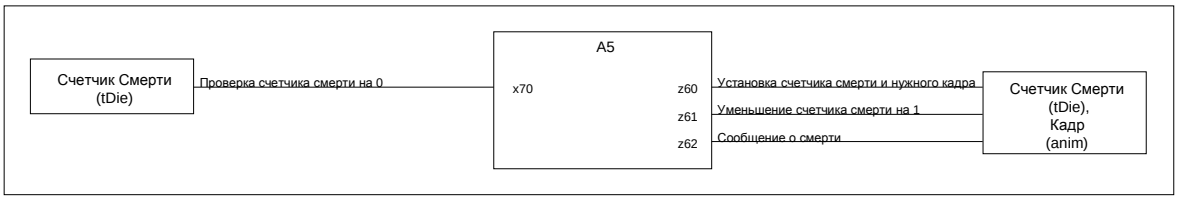

Рис. 28. Схема связей автомата анимации смерти (A5) класса «Монстр»

11.8.3. Граф переходов

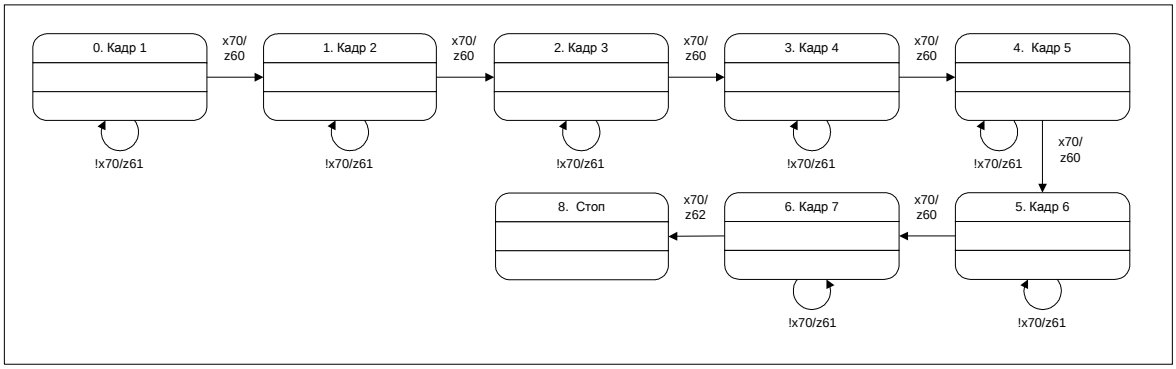

Рис. 29. Граф переходов автомата анимации смерти (A5) класса «Монстр»

12. Класс «Бомба» (CBomb)

12.1. Словесное описание

Задача этого класса состоит в управлении бомбами в игре. Класс хранит данные о бомбе (таймер, размер взрыва, положение на экране и т.д.).

12.2. Структурная схема класса

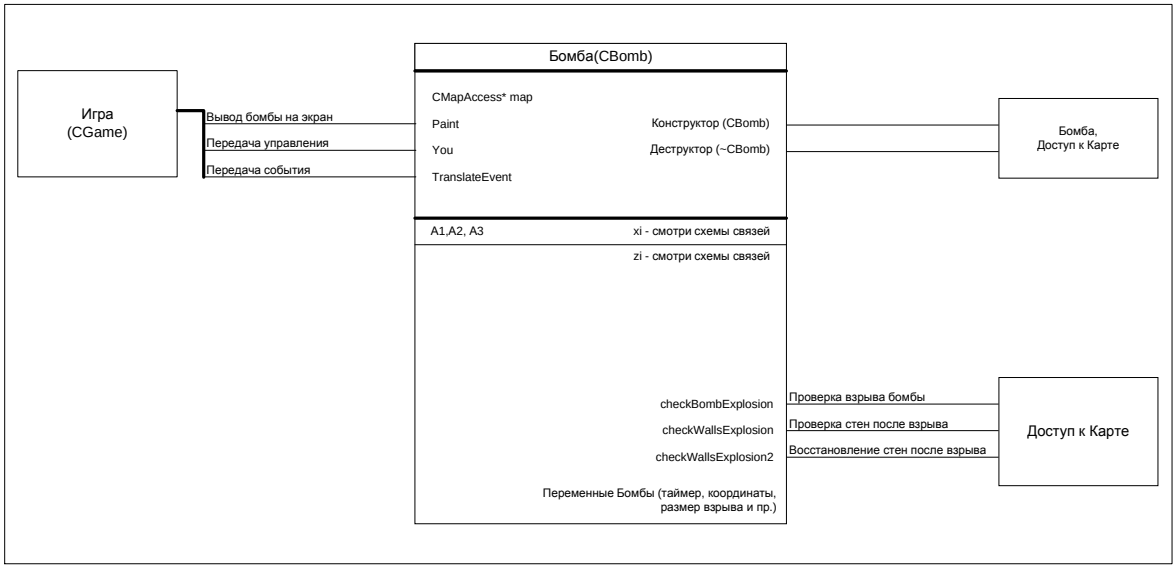

Рис. 30. Структурная схема класса «Бомба»

12.3. Схема взаимодействия автоматов

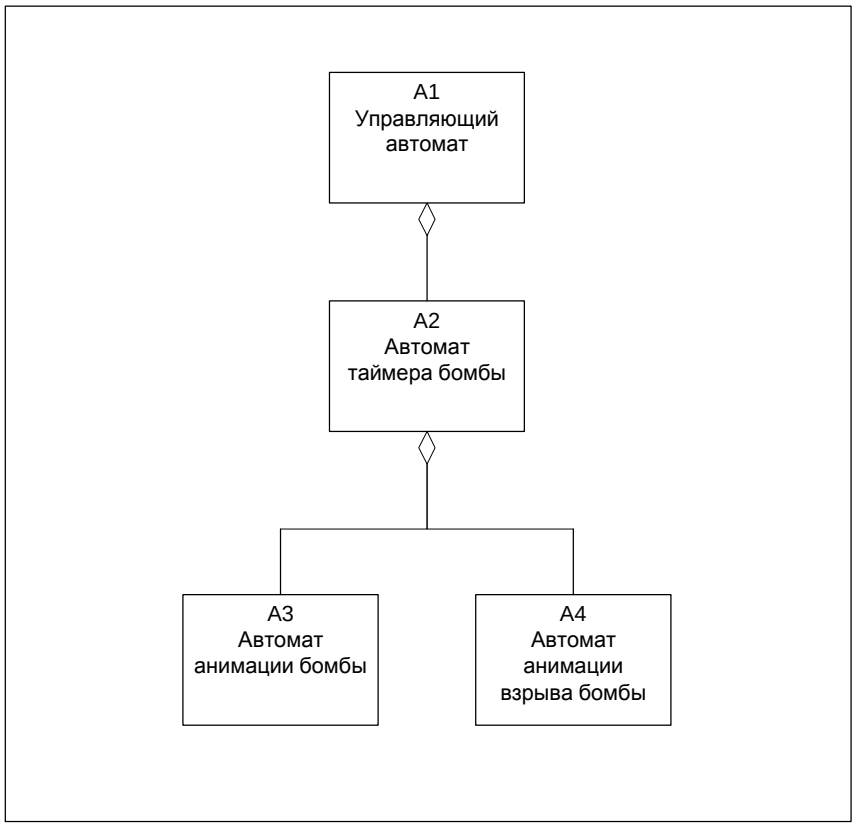

Рис. 31. Схема взаимодействия автоматов класса «Бомба»

12.4. Управляющий автомат (A1)

12.4.1. Словесное описание

Автомат является главным в классе «Бомба». Он вызывает другие автоматы в этом классе и обрабатывает события. После установки бомбы ее таймер начинает отсчитывать время. Через установленное время бомба взрывается.

12.4.2. Схема связей

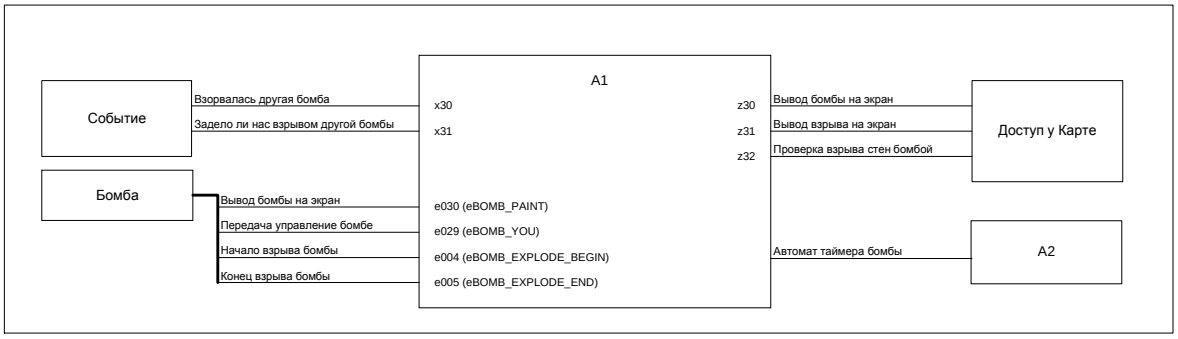

Рис. 32. Схема связей управляющего автомата (A1) класса «Бомба»

12.4.3. Граф переходов

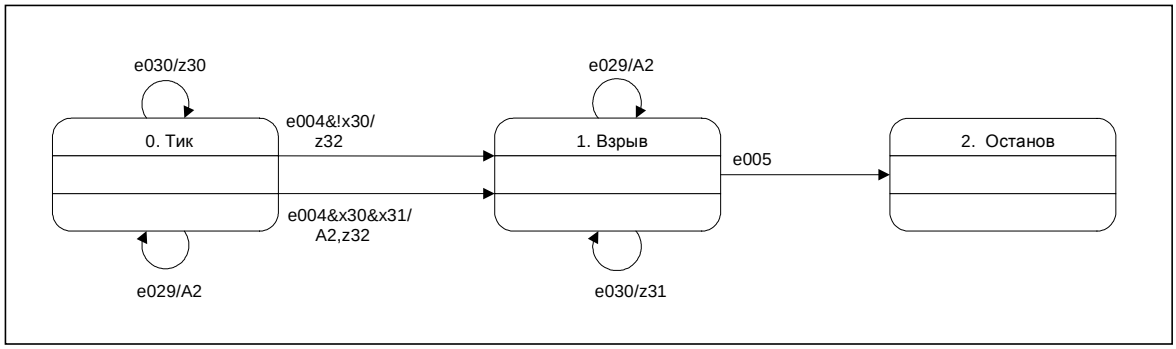

Рис. 33. Граф переходов управляющего автомата (A1) класса «Бомба»

12.5. Автомат таймера бомбы (A2)

12.5.1. Словесное описание

После установки бомбы ее таймер начинает отсчитывать время до взрыва. Этим процессом управляет данный автомат.

12.5.2. Схема связей

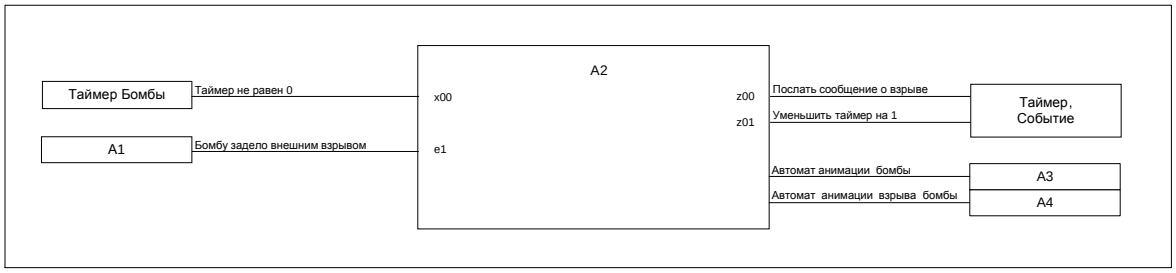

Рис. 34. Схема связей автомата таймера бомбы (A2) класса «Бомба»

12.5.3. Граф переходов

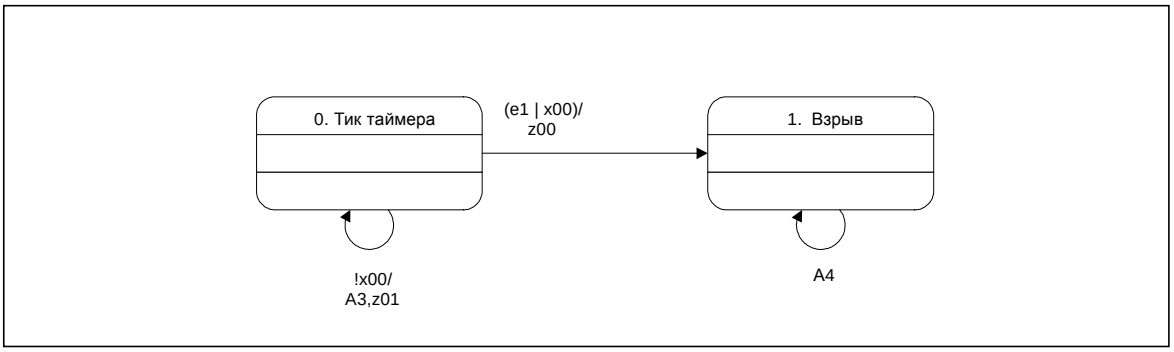

Рис. 35. Граф переходов автомата таймера бомбы (A2) класса «Бомба»

12.6. Автомат анимации бомбы (A3)

12.6.1. Словесное описание

Автомат управляет сменой кадров бомбы для создания эффекта ее пульсации.

12.6.2. Схема связей

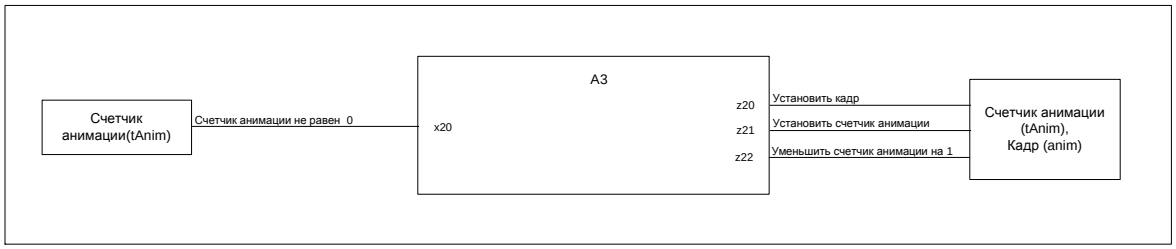

Рис. 36. Схема связей автомата анимации (A3) класса «Бомба»

12.6.3. Граф переходов

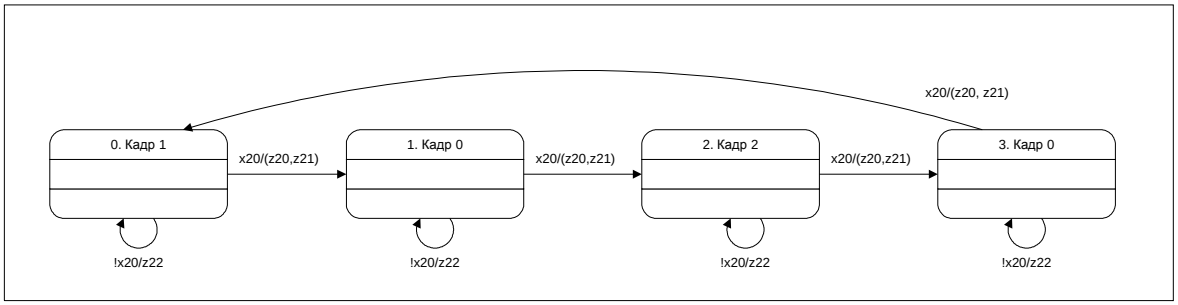

Рис. 37. Граф переходов автомата анимации (А3) класса «Бомба»

12.7. Автомат анимации взрыва бомбы (А4)

12.7.1. Словесное описание

Когда таймер бомбы сработал или ее задел взрыв другой бомбы, она начинает взрываться. Огонь после мгновеного распространения начинает медленно затухать. Этим процессом управляет автомат анимации взрыва бомбы.

12.7.2. Схема связей

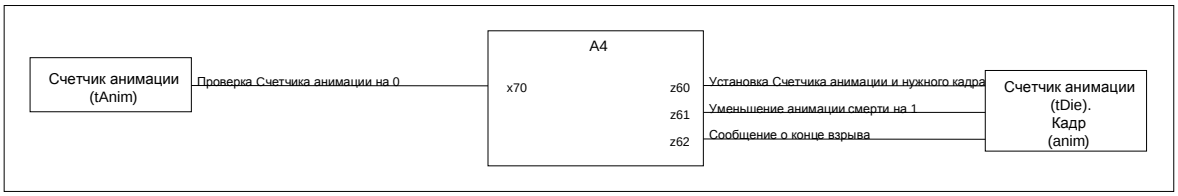

Рис. 38. Схема связей автомата анимации взрыва (А4) класса «Бомба»

12.7.3. Граф переходов

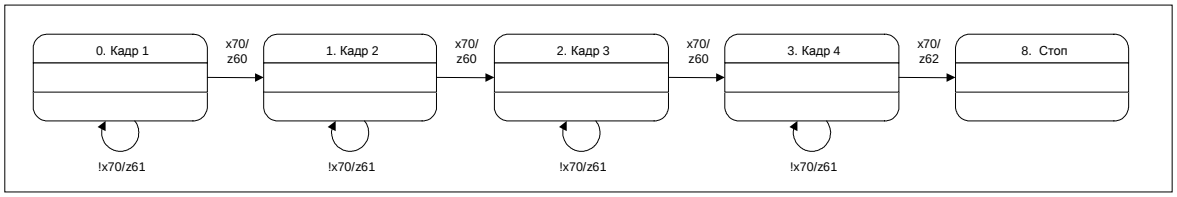

Рис. 39. Граф переходов автомата анимации взрыва (А4) класса «Бомба»

13. Реализация

Исходные тексты и сама программа приведены на сайте http://is.ifmo.ru и доступны из аннотации к настоящей работе. В приложении 2 приведена программная реализация всех автоматов.

Список событий

Событие	Определение	Название
e000	eKEY_DOWN	Нажата клавиша
e001	eKEY_UP	Отпущена клавиша
e002	ePLAYER_KEY_DOWN	Игрок нажал клавишу
e003	ePLAYER_KEY_UP	Игрок отпустил клавишу
e004	eBOMB_EXPLODE_BEGIN	Бомба начала взрываться
e005	eBOMB_EXPLODE_END	Бомба закончила взрываться
e006	eMONSTER_DIE_END	Монстр закончил умирать
e007	ePLAYER_DIE_END	Персонаж закончил умирать
e008	EYOU	Передача управления всем объектам на поле
e009	EPAINT	Отрисовка всего экрана
e010	eGAME_OVER	Конец игры
e011	eMONSTER_DIE_BEGIN	Монстр начал умирать
e012	eGAME_KEY_DOWN	Нажата клавиша в игре
e013	eGAME_KEY_UP	Отпущена клавиша в игре
e014	eSTART_GAME	Начало игры
e015	ePLAYER_YOU	Передача управление персонажу
e016	eBOMB_STOP_WALL	Бомба наскочила на стенку
e017	ePLAYER_DIE_BEGIN	Персонаж начал умирать
e018	ePLAYER_HIT	Персонажа ранили
e019	ePLAYER_PAINT	Отрисовка персонажа
e020	eMONSTER_HIT	Монстр ранен

e021	eMONSTER_PAINT	Отрисовка монстра
e022	eMONSTER_YOU	Передача управление монстру
e023	eKEY_DOWN (VK_LEFT)	Нажатие клавиши «Влево»
e024	eKEY_DOWN (VK_DOWN)	Нажатие клавиши «Вниз»
e025	eKEY_DOWN (VK_RIGHT)	Нажатие клавиши «Вправо»
e026	eKEY_DOWN (VK_UP)	Нажатие клавиши «Вверх»
e027	eKEY_DOWN (VK_SPACE)	Нажатие клавиши «Пробел»
e028	eKEY_DOWN (VK_ESCAPE)	Нажатие клавиши «Esc»
e029	eBOMB_YOU	Передача управление бомбе
e030	eBOMB_PAINT	Отрисовка бомбы
e999	eNO_EVENT	Зарезервированное сабытие

Листнинг программных реализаций автоматов

```
//Управляющий автомат (A0) класса «Игра»

void CGame::A0(CEvent *e)
{
    int i;
    switch(Y0)
    {
    case 0:
        if (e000(e)|| e001(e)){
            z01(e);
        }
        else if (e002(e)|| e003(e)){
            z02(e);
        }
        else if (e004(e)){
            z03(e);
        }
        else if (e005(e)){
            z04(e);
        }
        else if (e006(e)){
            z05(e);
        }
        else if (e007(e)){
            z06(e);
        }
        else if (e008(e)){
            z07(e);z11();
        }
        else if (e009(e)){
            z08(e);
        }
        else if (e010(e)){
            z09(e);
            Y0 = 1;
        }
        break;

    case 1:
        if (e024(e)){
            Y0 = 2;
        }
        else if (e027(e)){
            Y0 = 0;
        }
        else if (e028(e)){
            z10();
        }
        else if (e009(e)){
            z00(1);
        }
        break;
```

```
    case 2:
        if (e024(e)){
            Y0 = 3;
        }
        else if (e026(e)){
            Y0 = 1;
        }
        else if (e028(e)){
            z10();
        }
        else if (e009(e)){
            z00(2);
        }
        break;

    case 3:
        if (e024(e)){
            Y0 = 4;
        }
        else if (e026(e)){
            Y0 = 2;
        }
        else if (e027(e)){
            Y0 = 5;
        }
        else if (e028(e)){
            z10();
        }
        else if (e009(e)){
            z00(3);
        }
        break;

    case 4:
        if (e026(e)){
            Y0 = 3;
        }
        else if (e027(e)){
            z10();
        }
        else if (e028(e)){
            z10();
        }
        else if (e009(e)){
            z00(4);
        }
        break;

    case 5:
        if (e009(e)){
            z00(5);
        }
        else if (e028(e)){
            Y0 = 3;
        }
        break;
    }
}
```

```
//Управляющий автомат (A1) в классе «Игрок»

void CPlayer::A1(CEvent *e)
{
    int i;
    switch (Y1)
    {
    case 0:
        if (e019(e)){
            z00(e);
            z07(e);
        } else if (e015(e)){
            A3();
            z01(e);
        } else if (e002(e)){
            z02(e);
        } else if (e003(e)){
            z03(e);
        } else if (e005(e)){
            z04(e);
        } else if (e004(e) && !(x50(e) && !x51())){
            z09(e);
        } else if (e004(e) && x50(e) && x51()){
            z09(e);
            z05(e);
            Y1 = 2;
        } else if (e004(e) && x50(e) && !x51()){
            z09(e);
            z19(e);
            Y1 = 1;
        }
        break;

    case 1:
        if (e015(e)){
            A5();
            z01(e);
        } else if (e019(e)){
            z00(e);
        } else if (e007(e)){
            Y1 = 3;
        } else if (e005(e)){
            z04(e);
        } else if (e004(e)){
            z09(e);
        }
        break;

    case 2:
        if (e002(e)){
            z02(e);
        } else if (e003(e)){
            z03(e);
        } else if (e015(e) && !x53()){
            A3();
            A4();
            z01(e);
            z10();
        } else if (e015(e) && x53()){
            A3();
            z01(e);
```

```
            Y1 = 0;
        } else if (e019(e) && x52()){
            z00(e);
        } else if (e004(e)){
            z09(e);
        } else if (e005(e)){
            z04(e);
        }
        break;

    case 3:
        if (e015(e)){
            z01(e);
        } else if (e019(e)){
            z07(e);
        } else if (e004(e)){
            z09(e);
        } else if (e005(e)){
            z04(e);
        }
        break;
    }
}


//Автомат анимации (A2) в классе «Игрок»

void CPlayer::A2()
{
    switch(Y2)
    {
    case 0:
        if (x00()){
            z20(1);
            z21();
            Y2 = 1;
        } else if (!x00()){
            z22();
        }
        break;

    case 1:
        if (x00()){
            z20(0);
            z21();
            Y2 = 2;
        } else if (!x00()){
            z22();
        }
        break;

    case 2:
        if (x00()){
            z20(2);
            z21();
            Y2 = 3;
        } else if (!x00()){
            z22();
        }
        break;
```

```
    case 3:
        if (x00()){
            z20(0);
            z21();
            Y2 = 0;
        } else if (!x00()){
            z22();
        }
        break;
    }
}


//Автомат управления движением (A3) в классе «Игрок»

void CPlayer::A3()
{
    switch(Y3)
    {
    case 0:
        if (x11()){
            z31(1,-1);
            z30();
            Y3 = 1;
        }
        if (x12()){
            z31(2,1);
            z30();
            Y3 = 2;
        }
        if (x13()){
            z31(1,1);
            z30();
            Y3 = 3;
        }
        if (x14()) {
            z31(2,-1);
            z30();
            Y3 = 4;
        }
        break;

    case 1:
        A2();
        if (!x17() && x29() || x10()){
            dir = 1;
            Y3 = 0;
        }
        else if (x29() && x20() && x24()){
            z31(2,-1);
            z30();
            z32();
            Y3 = 4;
        }
        else if (x29() && x18() && x22()){
            z31(2,1);
            z30();
            z32();
            Y3 = 2;
        }
```

```
        else if (x19() && x23()){
            z31(1,1);
            z30();
            z32();
            Y3 = 3;
        }
        else{
            z31(1,-1);
            z30();
        }
        break;

    case 2:
        A2();
        if (!x18() && x30()|| x10()){
            dir = 2;
            Y3 = 0;
        }
        else if (x30() && x17() && x21()){
            z31(1,-1);
            z30();
            z33();
            Y3 = 1;
        }
        else if (x30() && x19() && x23()){
            z31(1,1);
            z30();
            z33();
            Y3 = 3;
        }
        else if (x20() && x24()){
            z31(2,-1);
            z30();
            z33();
            Y3 = 4;
        }
        else{
            z31(2,1);
            z30();
        }
        break;

    case 3:
        A2();
        if (!x19() && x29() || x10()){
            dir = 3;
            Y3 = 0;
        }
        else if (x29() && x18() && x22()){
            z31(2,1);
            z30();
            z34();
            Y3 = 2;
        }
        else if (x29() && x20() && x24()){
            z31(2,-1);
            z30();
            z34();
            Y3 = 4;
        }
```

```
        else if (x17() && x21()){
            z31(1,-1);
            z30();
            z34();
            Y3 = 1;
        }
        else {
            z31(1,1);
            z30();
        }
        break;

    case 4:
        A2();
        if (!x20() && x30() || x10()){
            dir = 4;
            Y3 = 0;
        }
        else if (x30() && x19() && x23()){
            z31(1,1);
            z30();
            z35();
            Y3 = 3;
        }
        else if (x30() && x17() && x21()){
            z31(1,-1);
            z30();
            z35();
            Y3 = 1;
        }
        else if (x18() && x22()){
            z31(2,1);
            z30();
            z35();
            Y3 = 2;
        }
        else {
            z31(2,-1);
            z30();
        }
        break;
    }
}


//Автомат мигания (A4) в классе «Игрок»

void CPlayer::A4()
{
    switch(Y4)
    {
    case 0:
        if (!x60()){
            z50();
        }
        else if (x60()){
            z51(0);
            Y4 = 1;
        }
        break;
```

```
    case 1:
        if(!x60()){
            z50();
        }
        else if (x60()) {
            z51(1);
            Y4 = 0;
        }
        break;
    }
}


//Автомат анимации смерти (A5) в классе «Игрок»

void CPlayer::A5()
{
    switch(Y5)
    {
    case 0:
        if (x70()){
            Y5 = 1;
        }
        else{
            z61();
        }
        break;

    case 1:
        if (x70()){
            z60(2);
            Y5 = 2;
        }
        else{
            z61();
        }
        break;

    case 2:
        if (x70()){
            z60(3);
            Y5 = 3;
        }
        else{
            z61();
        }
        break;

    case 3:
        if (x70()){
            z60(4);
            Y5 = 4;
        }
        else{
            z61();
        }
        break;
```

```
    case 4:
        if (x70()){
            z60(5);
            Y5 = 5;
        }
        else{
            z61();
        }
        break;

    case 5:
        if (x70()){
            z60(6);
            Y5 = 6;
        }
        else{
            z61();
        }
        break;

    case 6:
        if (x70()){
            z62();
            Y5 = 8;
        }
        else{
            z61();
        }
        break;

    case 8:
        break;
    }
}


//Управляющий автомат (A1) в классе «Монстр»

void CMonster::A1(CEvent *e)
{
    int i;
    switch (Y1)
    {
    case 0:
        if (e021(e)) {
            z00(e);
        }
        else if (e022(e)){
            A3();
        }
        else if (e004(e) && x50(e) && x51()){
            z05(e);
            Y1 = 2;
        }
        else if (e004(e) && x50(e) && !x51()){
            z19(e);
            Y1 = 1;
        }
        break;
```

```
    case 1:
        if  (e022(e)){
            A5();
        }
        else if (e021(e)){
            z00(e);
        }
        else if (e006(e)){
            Y1 = 3;
        }
        break;

    case 2:
        if  (e022(e) && !x53()){
            A3();
            A4();
            z10();
        }
        else if (e022(e) && x53()){
            A3();
            Y1 = 0;
        }
        else if (e021(e)){
            z00(e);
        }
        break;

    case 3:
        break;
    }
}


//Автомат анимации (A2) в классе «Монстр»

void CMonster::A2()
{
    switch(Y2)
    {
    case 0:
        if  (x00()) {
            z20(1);
            z21();
            Y2 = 1;
        }
        else if (!x00()){
            z22();
        }
        break;

    case 1:
        if  (x00()) {
            z20(0);
            z21();
            Y2 = 2;
        }
        else if (!x00()){
            z22();
        }
        break;
```

```
    case 2:
        if  (x00()) {
            z20(2);
            z21();
            Y2 = 3; }
        else if (!x00()){
            z22();
        }
        break;

    case 3:
        if  (x00()) {
            z20(0);
            z21();
            Y2 = 0;
        }
        else if (!x00()){
            z22();
        }
        break;
    }
}


//Автомат управления движением (A3) в классе «Монстр»

void CMonster::A3()
{
    switch(Y3)
    {
    case 0:
        if (x09()){
            Y3 = 1;
        }
        else{
            z32();
            A2();
        }
        break;

    case 1:
        if (x08())
        {
            z30();
            z31();
            Y3 = 0;
        }
        else
        {
            z30();
        }
        break;
    }
}
```

```
//Автомат мигания (A4) в классе «Монстр»

void CMonster::A4()
{
    switch(Y4)
    {
    case 0:
        if  (!x60()){
            z50();
        }
        else if (x60()) {
            z51(false);
            Y4 = 1;
        }
        break;

    case 1:
        if  (!x60()){
            z50();
        }
        else if (x60()) {
            z51(true);
            Y4 = 0;
        }
        break;
    }
}


//Автомат анимации смерти (A5) в классе «Монстр»

void CMonster::A5()
{
    switch(Y5)
    {
    case 0:
        if (x70()){
            Y5 = 1;
        }
        else{
            z61();
        }
        break;

    case 1:
        if (x70()){
            z60(2);
            Y5 = 2;
        }
        else{
            z61();
        }
        break;
```

```
    case 2:
        if (x70()){
            z60(3);
            Y5 = 3;
        }
        else{
            z61();
        }
        break;

    case 3:
        if (x70()){
            z60(4);
            Y5 = 4;
        }
        else{
            z61();
        }
        break;

    case 4:
        if (x70()){
            z60(5);
            Y5 = 5;
        }
        else{
            z61();
        }
        break;

    case 5:
        if (x70()){
            z60(6);
            Y5 = 6;
        }
        else{
            z61();
        }
        break;

    case 6:
        if (x70()){
            z62();
            Y5 = 8;
        }
        else{
            z61();
        }
        break;

    case 8:
    break;
    }
}
```

```
//Управляющий автомат (A1) в классе «Бомба»

void CBomb::A1(int e)
{
    switch (Y1)
    {
        case 0:
            if (e == 1 || x00()){
                z00();
                Y1 = 1;
            }
            else if (!x00()){
                A2();
                z01();
            }
            break;

        case 1:
            A3();
            break;
    }
}


//Автомат таймера бомбы (A2) в классе «Бомба»

void CBomb::A2()
{
    switch(Y2)
    {
    case 0:
        if  (x20()) {
            z20(1);
            z21();
            Y2 = 1;
        }
        else if (!x20()){
            z22();
        }
        break;

    case 1:
        if  (x20()) {
            z20(0);
            z21();
            Y2 = 2;
        }
        else if (!x20()){
            z22();
        }
        break;
```

```
    case 2:
        if  (x20()) {
            z20(2);
            z21();
            Y2 = 3;
        }
        else if (!x20()){
            z22();
        }
        break;

    case 3:
        if  (x20()) {
            z20(0);
            z21();
            Y2 = 0;
        }
        else if (!x20()){
            z22();
        }
        break;
    }
}


//Автомат анимации бомбы (A3) в классе «Бомба»

void CBomb::A3()
{
    switch(Y3)
    {
    case 0:
        if (x70()){
            Y3 = 1;
        }
        else{
            z61();
        }
        break;

    case 1:
        if (x70()){
            z60(2);
            Y3 = 2;
        }
        else {
            z61();
        }
        break;

    case 2:
        if (x70()){
            z60(3);
            Y3 = 3;
        }
        else{
            z61();
        }
        break;
```

```
    case 3:
        if (x70()){
            z62();
            Y3 = 8;
        }
        else {
            z61();
        }
        break;

    case 8:
        break;
    }
}
```