Not Yet Another Cognitive Actors Team Survival

(NYA CATS)

## Общая характеристика.

Имеется прямоугольное клетчатое поле, на нём находятся няки, которые разбиты на две и более противоборствующих команд. У каждой няки в начале игры есть специализация и соответствующие ей цели и способности. Цель команды в целом - уничтожить противника. Также по полю разбросаны всякие разные плюшки, которые няки могут использовать для разных штук, например, мясо.

***Основные понятия.***

* *Игровой объект* — однозначно идентифицируемая сущность в игровом пространстве-времени
* *Активный игровой объект* — игровой объект, имеющий возможность самостоятельно инициировать взаимодействие с другими объектами и/или между другими объектами
* *Пассивный игровой объект* — игровой объект, имеющий возможность взаимодействовать с другими объектами, но не могущий это взаимодействия инициировать
* *Игровая единица (персонаж)* — конкретный активный игровой объект, имеющий в каждый момент времени определённые значения характеристик, поведение (возможно отличающееся от первоначального) и положение в игровом пространстве, принадлежит определённому классу и определённой команде.
* *Класс игровых единиц* — совокупность игровых единиц, имеющих одинаковые начальные значения характеристик, набор способностей и собственное поведение, и уникальный способ визуализации.
* *Команда* — совокупность персонажей, каждый их который находится в дружеских отношениях с другими персонажами данной команды.
* Будем говорить, что два персонажа находятся в *дружеских* отношениях, если они не имеют своей целью принести какой-либо вред друг другу
* Будем говорить, что два персонажа находятся во *вражде*, если хотя бы один из них имеет своей целью (или вынужден для своего выживания) причинить вред другому.
* *Характеристика* — это свойство игрового объекта, определяющее качество и в некоторых случаях множество способов его взаимодействия с другими объектами, для конкретного объекта имеет определённую степень проявления, в процессе этих взаимодействий могут быть задействованы другие объекты.
* *Способность* — это свойство игрового объекта, определяющее множество способов его взаимодействия с другими объектами, имеет определённую степень проявления, в процессе этих взаимодействий могут быть задействованы другие объекты.
* *Окружающая среда* — активный игровой объект, имеющий протяжённость в игровом времени и пространстве, в области действия которого находятся все остальные игровые объекты
* *Игровое поле* — пассивная часть окружающей среды, состоит из клеток и имеет прямоугольную форму

*Структура*

*Цели (выразить)*

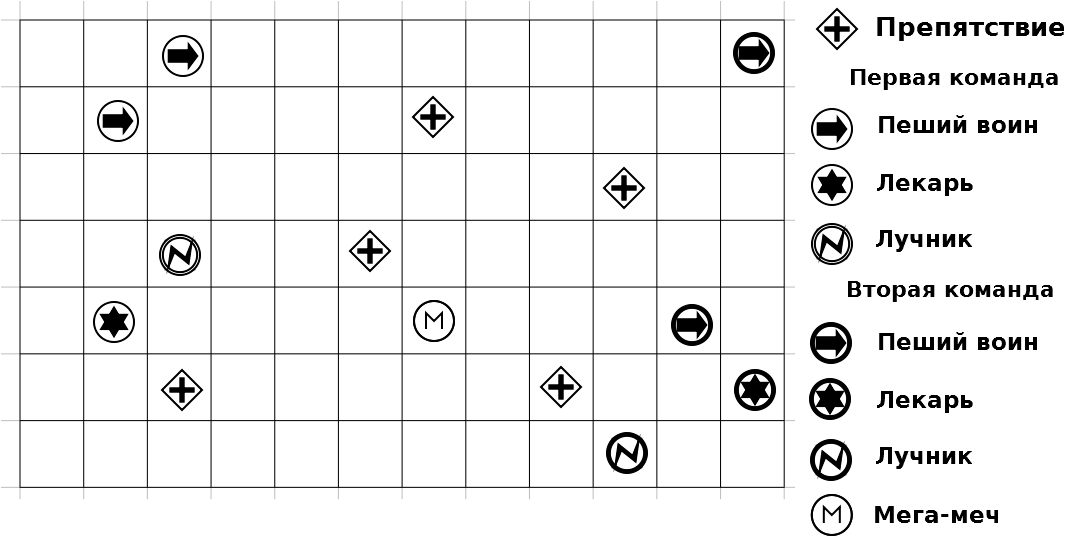
*Домен действий*

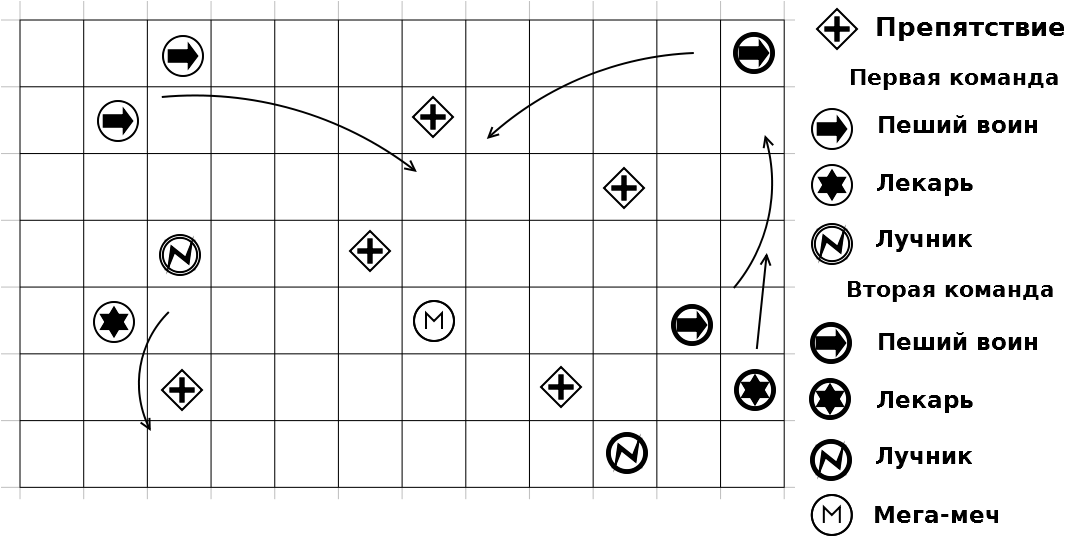
*Двигаться сторону члена команды, обойти препятствие, захватить ресурс*

*Названия ситуаций, приоритеты действия для ситуаций, командная оценка ситуации*

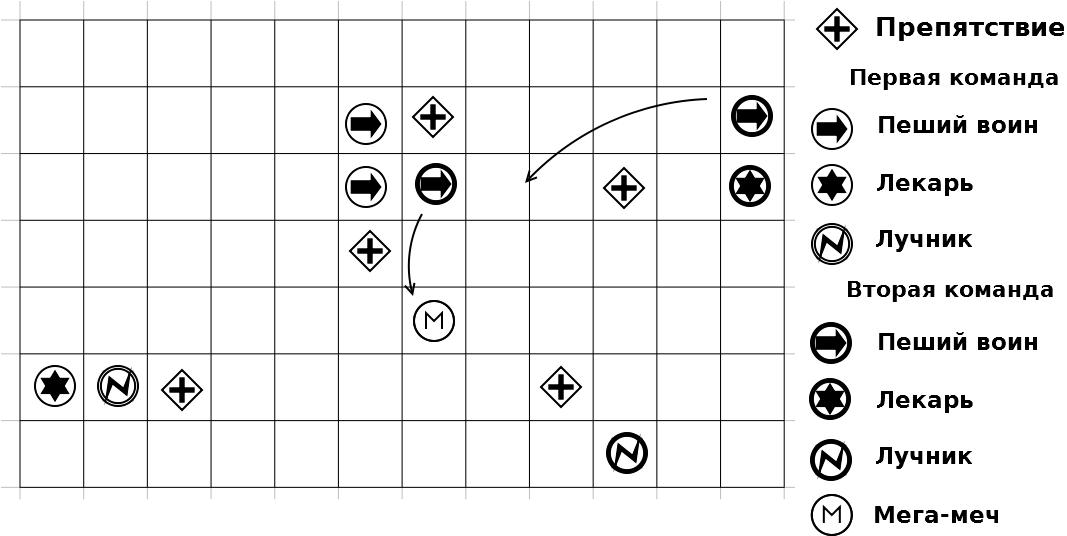
*Видимость больше!*

***Пример***

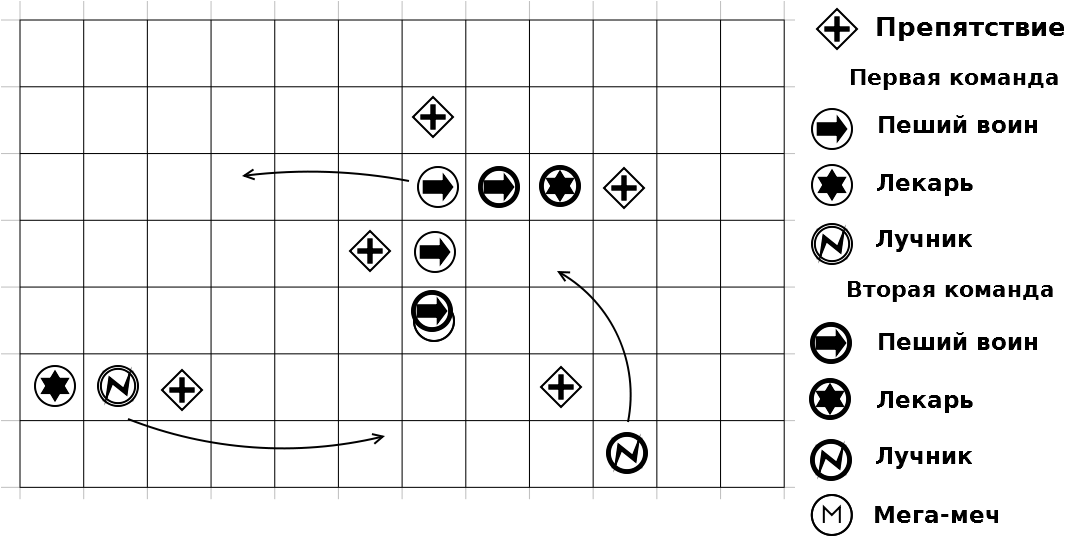


Поскольку у каждого персонажа есть определённая область видимости (в данном случае предположим, что персонаж видит только соседние клетки), то в два воина первой команды, увидев друг друга, сразу скооперируются, как и лучник и лекарь, а во второй команде скооперируются только воин и лекарь, при этом лекарь идёт позади воина, что вполне естественно. 

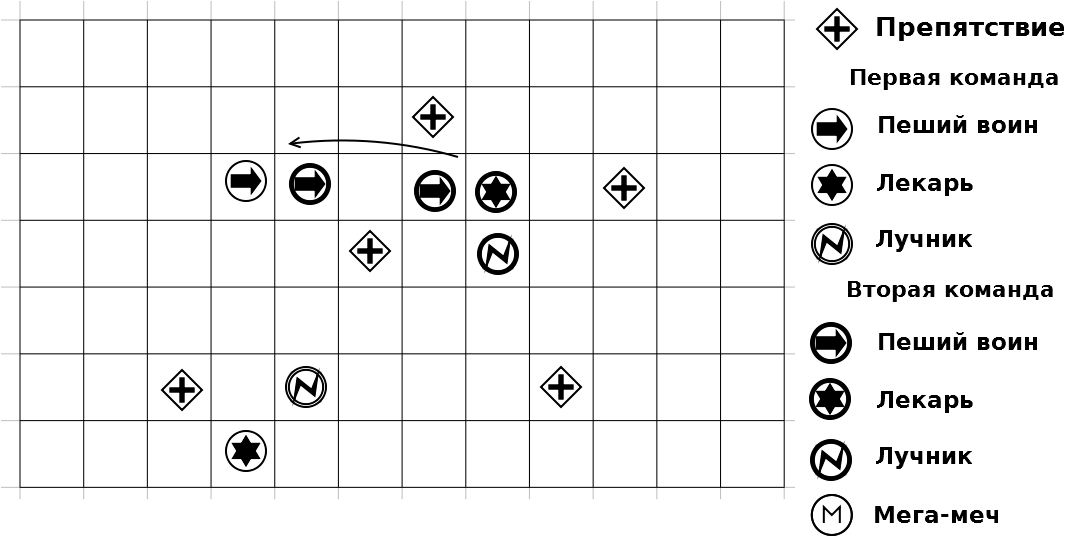
Лучник первой команды узнаёт у лекаря, что он видит камень, и решает за ним спрятаться. Воины первой команды решают идти вправо. Лучник второй команды остаётся прятаться за камнем, один из воинов решает идти вверх, и за ним идёт лекарь, другой воин идёт влево.



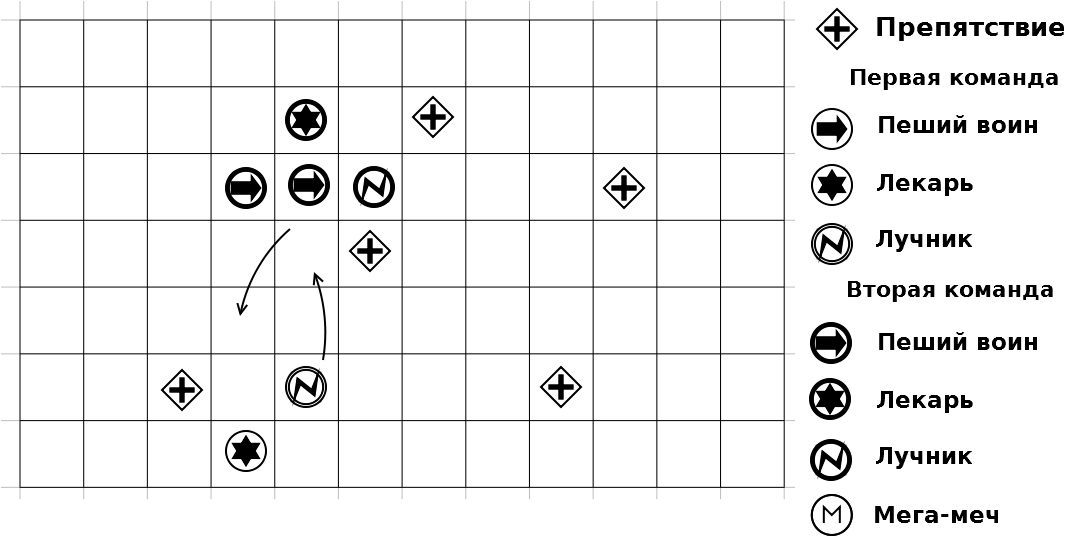
Два воина первой команды и один воин второй встречаются примерно в центре, и, получив один удар мечом и сделав ответный, воин второй команды решает отступать вниз. Воин и лекарь первой команды решают продолжать движение и идут влево.



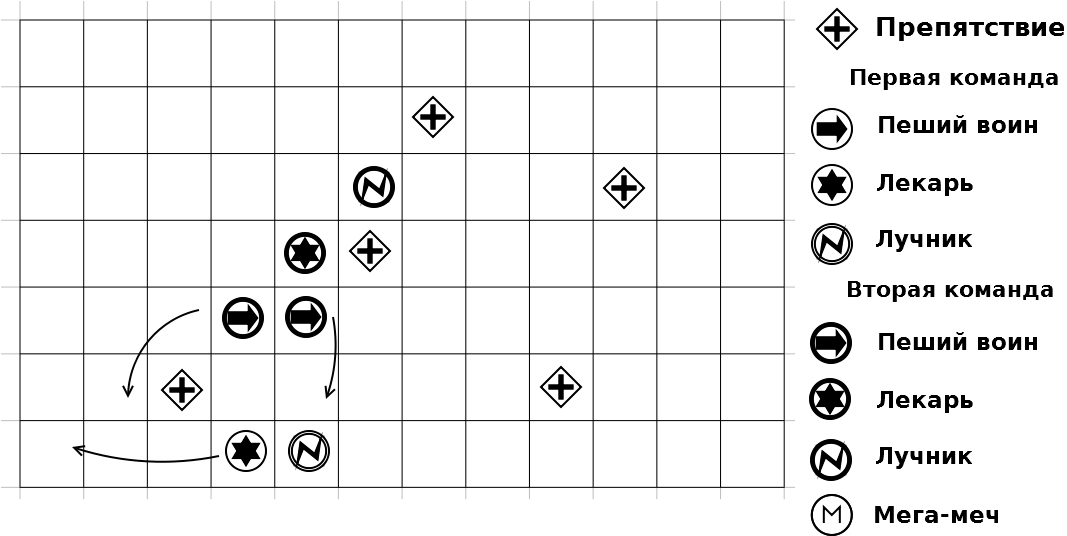
Лучник и лекарь первой команды решают идти в разведку вправо, а лучник второй команды - вверх. Тем временем лекарь второй команды залечивает воина.



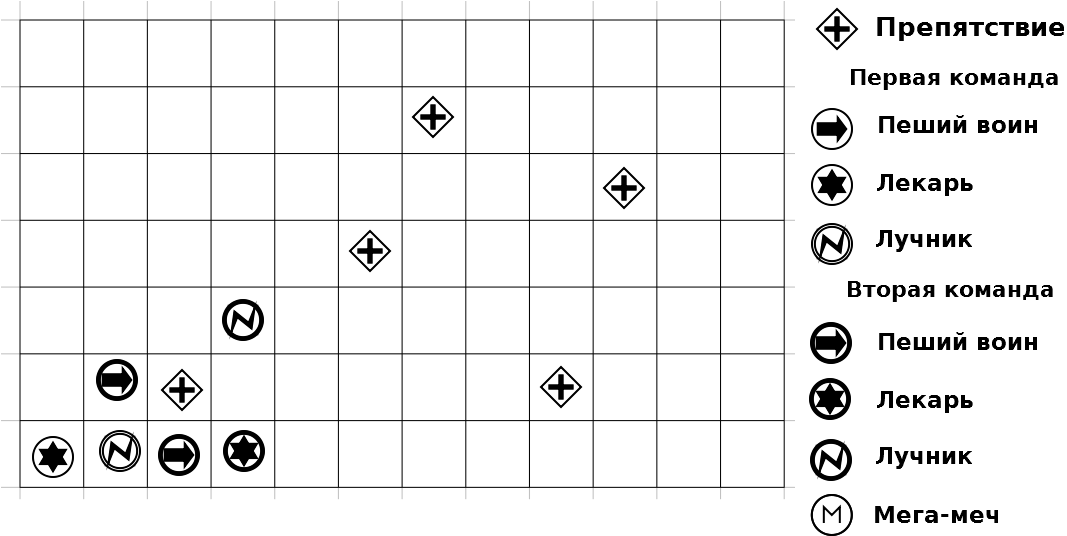
Вторая команда продолжает преследование воина первой команды, и ближайший к нему воин второй команды убивает его.



Лучник замечает вражеских воинов, начинает их обстреливать, но воины второй команды почти сразу лечатся лекарем, и начинают преследование вражеского лучника.



Лучник и лекарь первой команды продолжают отступать, а воины второй команды решают обойти с разных сторон препятствия.



Зажатые в угол игроки первой команды погибают. Вторая команда победила.

***Домены для рассмотренного примера***

Класс персонажей

* Воин
* Лучник
* Лекарь

Другие объекты

* Препятствия (камни)

Способности персонажей (общие)

* Передвижение
* Зрение

Способности пешего воина

* Удар мечом

Способности лучника

* Выстрел стрелой из лука

Способности лекаря

* Лечить

Характеристики персонажей (общие)

* Уровень здоровья
* Радиус обзора

Характеристики воина

* Сила удара
* Уровень защиты

Характеристики лекаря

* Скорость лечения

Характеристики лучника

* Урон от стрельбы

Постановка задачи:

Разработать когнитивную игру, в которой игровые объекты (воин, лучник, лекарь и др.) передвигаются по клеточному игровому полю, воюют между собой, собирают еду. Поведение и состав игровых объектов определяется при помощи информации из базы знаний, построенной в рамках фреймовой парадигмы.

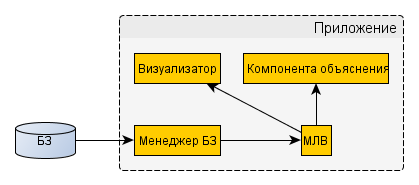
Модель

Есть игровое поле, состоящее из ячеек. В ячейках могут находиться игровые объекты: персонажи и др.

У персонажей изображение и разная яркость фона.

Для каждого персонажа определяется ситуация. Далее из ситуации берется действие и выполняется.

#### Архитектура приложения



**База знаний** (БЗ) хранит описание доменов, сети фреймов для представления игровых объектов, ситуаций, изображений и др., а также процедуры для реакции игровых персонажей на ситуации.

**Менеджер БЗ** управляет базой знаний. Он осуществляет загрузку БЗ из файла, преобразует ее во внутреннее представление. Обеспечивает все операции с БЗ, такие как загрузка БЗ, редактирование БЗ (добавление, изменение, удаление доменов, фреймов, слотов, процедур, продукций), сохранение БЗ. Также предоставляет доступ к БЗ для МЛВ.

**Механизм логического вывода** (МЛВ) необходим для осуществления логического вывода. Т.е. определения ситуаций, в которые попадают игровые персонажи, привязки картинок к игровым объектам, выполнения действий, указанных в процедурах, исполнение продукционных программ.

**Визуализатор** отображает ход игры, т.е. клеточное игровое поле и пиктограммы игровых объектов на нем. Отображает изменения в рабочей памяти, которые могли произойти во время работы МЛВ.

**Компонента объяснени** (КО) позволяет производить отладку БЗ, а именно смотреть содержимое рабочей памяти и ход вывода.

#### Фреймовая модель

База знаний состоит из 2 частей:

● Доменов

● Сети фреймов

В базе знаний присутствуют системные домены и фреймы, которые не могут быть удалены, так как это приведет к полной неработоспособности приложения.

В сети фреймов несколько типов фреймов:

● Игровые объекты

● Игровые ситуации

● Изображения объектов

У каждого фрейма есть по крайней мере 2 системных слота:

#name - имя фрейма

#is\_a - родитель фрейма.

У ситуаций дополнительно слот #action - действие, которое выполняется после привязки ситуации.

У изображений дополнительно:

#Картинка - слот, который содержит путь к изображению.

Используется наследование same, поэтому вывод идет сверху-вниз.

Для разнообразия поведения используется нетерпеливая стратегия(поиск первой привязавшейся ситуации) и случайный обход фреймов ситуаций.

МЛВ однопроходовый. Сразу после привязки ситуации выполняется действие. Таким образом решена проблема возникновения конфликтов. Картинка персонажа также определяется благодаря привязке.

Если не привязалась ни одна ситуация, то нечего делать - значит привязывается исключительлная ситуация нечего делать.

Если игровой объект окружен и некуда идти, то привязывается ситуация некуда идти. Это действие выполняется через API процедур.

БЗ имеет системную часть, которая должна быть обязательно:

Системные фреймы:

* Игровое поле
* Ячейка игрового поля
* Игровой объект
* Пусто
* Персонаж
* Изображение
* Ситуация
* Нечего делать
* Некуда идти

Системные домены:

* Команды

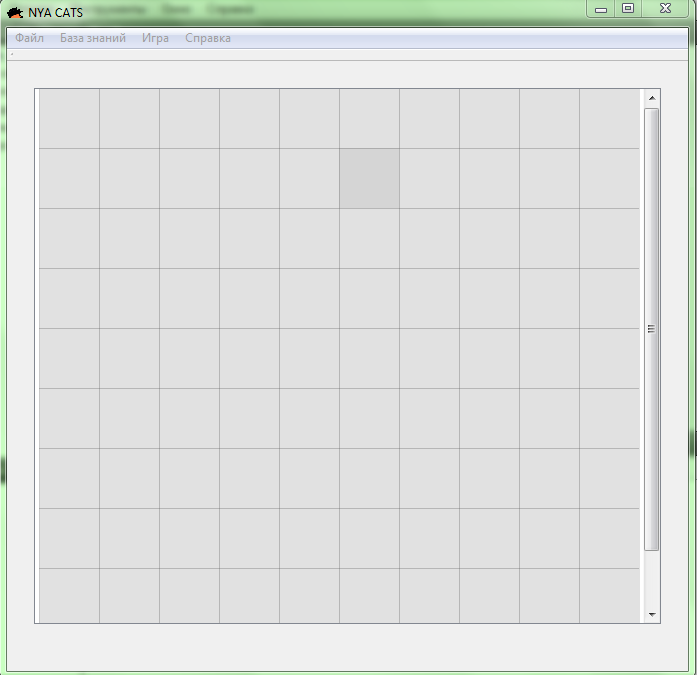
## Инструкция пользователя

### Интерфейс пользователя

#### Окна

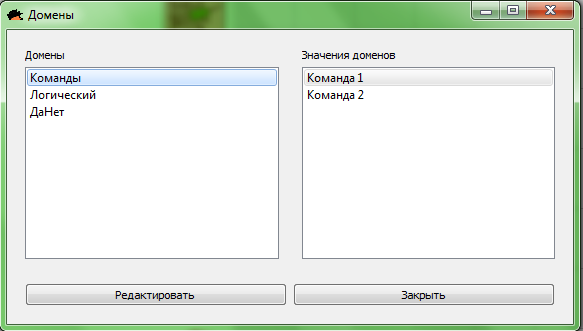
##### Главное окно

Содержит игровое поле, по которому передвигаются игровые объекты, а также после загрузки БЗ и игровые объекты, которые можно мышкой “кидать” на поле.



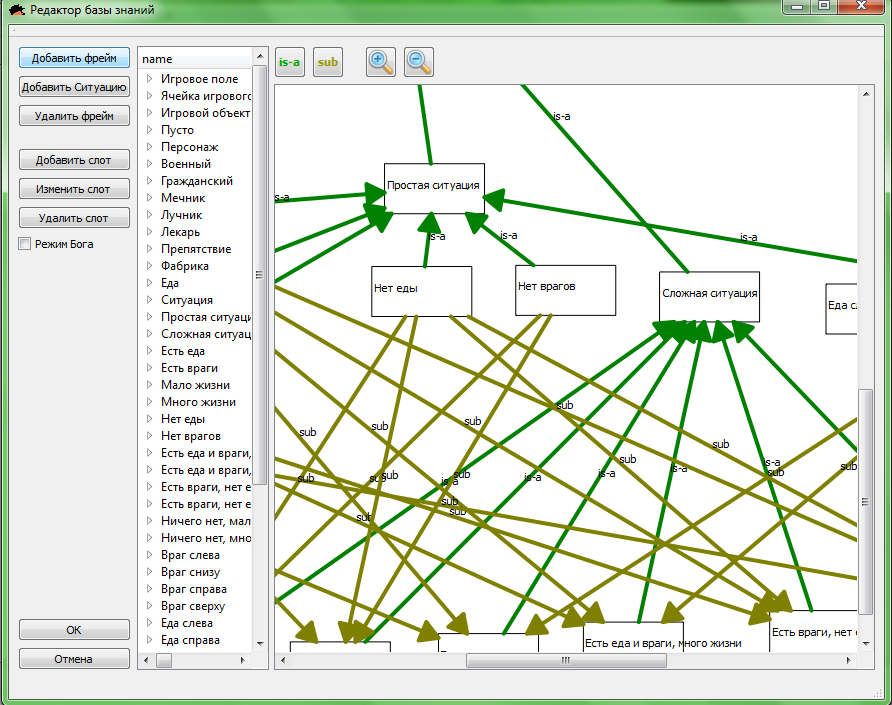
##### Окно редактора доменов

Позволяет добавлять и редактировать домены и их значения.



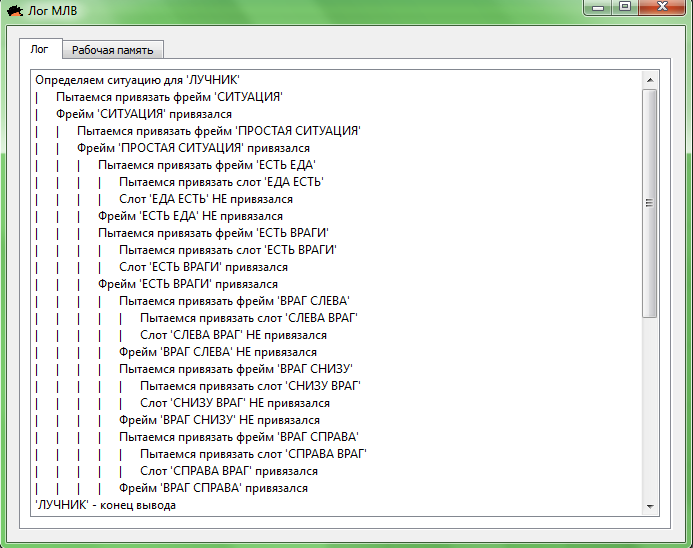
##### Окно редактора фреймов

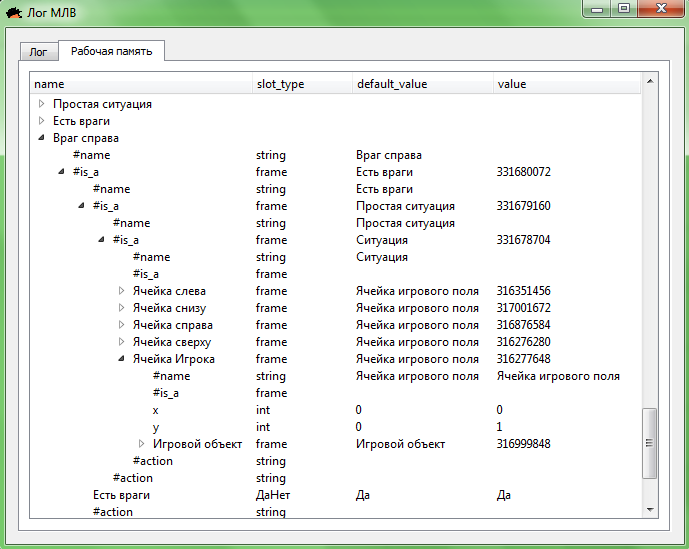
Позволяет добавлять редактировать фреймы базы знаний.

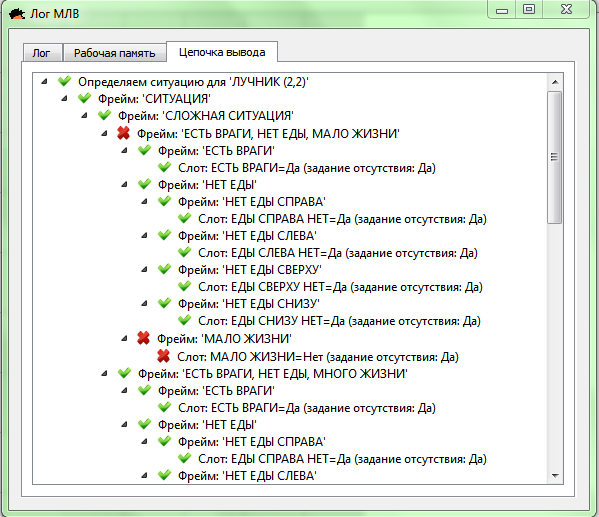


##### Окно отладочного вывода

Включает в себя 3 вкладки: лог МЛВ, визуализацию содержимого рабочей памяти и тривью для отображения хода привязки ситуаций и картинок персонажей.







#### Описание пунктов меню

**Файл:**

**Создать.** Создает новую БЗ. Менеджер БЗ создает пустую БЗ.

**Открыть.** Открывает существующую БЗ. Менеджер БЗ загружает файл с БЗ и преобразует БЗ из формата XML во внутреннее представление.

**Закрыть.** Закрывает текущую открытую БЗ. Менеджер БЗ удаляет БЗ.

**Сохранить.** Сохраняет изменения в БЗ. Менеджер БЗ сохраняет изменения в файл.

**Сохранить как...** Открывает диалог сохранения БЗ.

**Выход.** Завершает программу.

**База знаний:**

**Редактор фреймов.** Открывает редактор фреймов.

**Редактор доменов.** Открывает редактор доменов.

**Редактор глобальных процедур.** Открывает редактор процедур для редактирования ппроцедур, доступных глобально.

**Игра:**

**Начать игру.** Начинает игру. Дает команду для МЛВ начать игру.

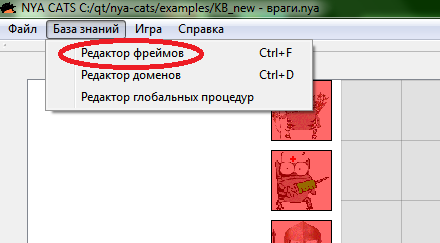
**Начать случайную игру.** Запускает игру со случайной расстановкой игровых объектов на поле. Дает команду МЛВ расставить случайное количество объектов на поле в случайных клетках, со случайным уровнем жизни в случае персонажей. Далее запускается игра.

**Сделать шаг.** Делает один шаг вывода на текущем поле. МЛВ производит вывод для всех персонажей.

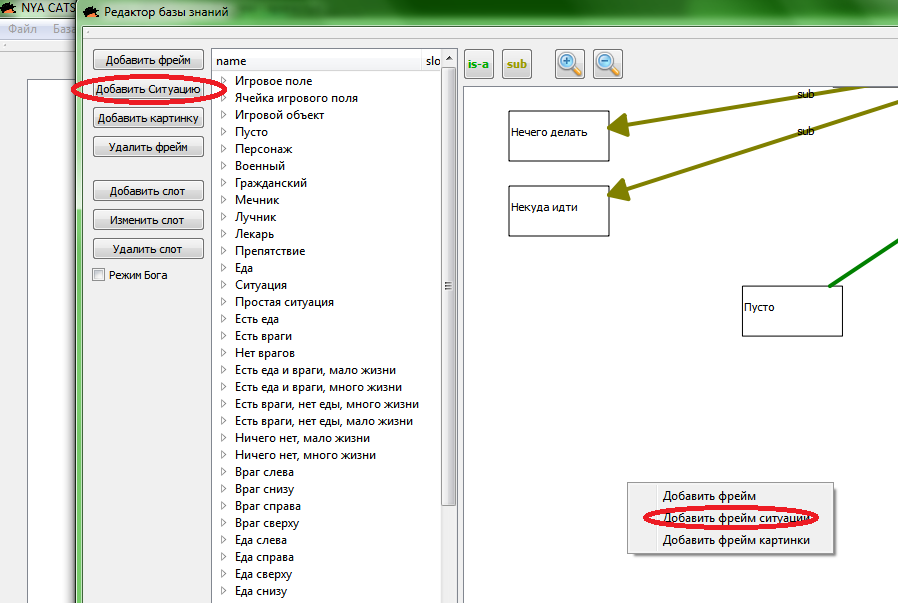
**Закончить игру.** Заканчивает игру.

#### Как добавить ситуацию

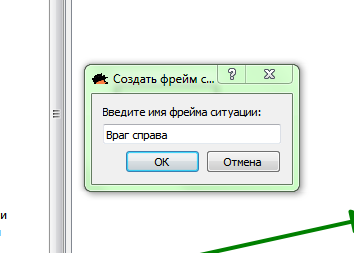
Открываем редактор фреймов через контекстное меню.



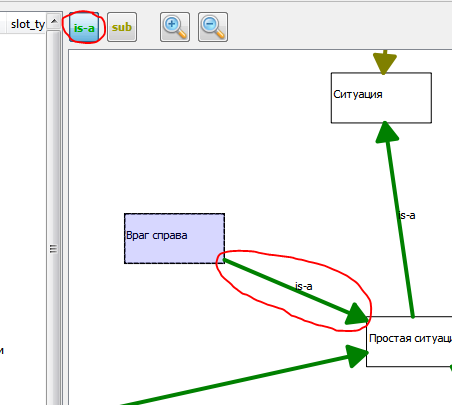
Добавляем фрейм ситуации, нажав соответствующую кнопку или через контекстное меню.



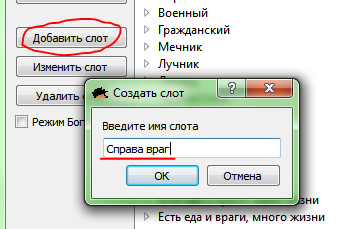
Задаем имя для нашей ситуации.



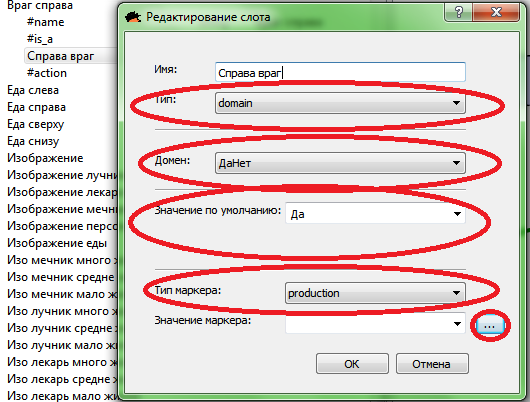
Протягиваем связь is\_a к фрейму “Ситуация” или к его потомку.



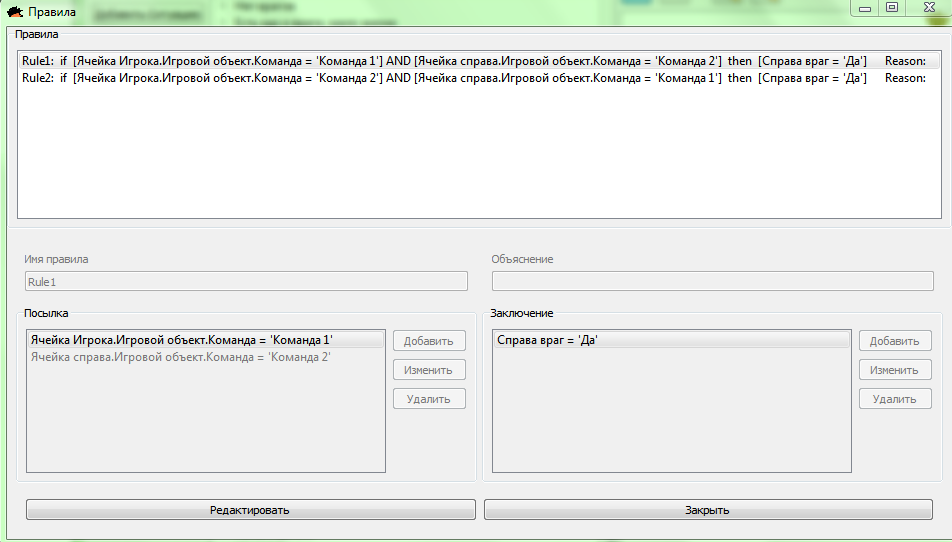
Выделяем созданный фрейм и добавляем слот “Справа враг”.



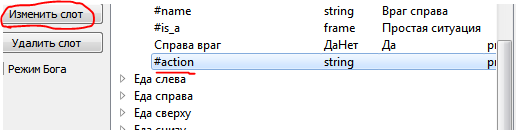
Появится окно редактирования слота. Выбираем тип “domain”. В поле “Домен” выбираем домен “ДаНет”. Проставляем значение по-умолчанию(задание отсутствия) “Да”. Выбираем тип маркера “production”. Добавляем продукционную программу для означивания слота.



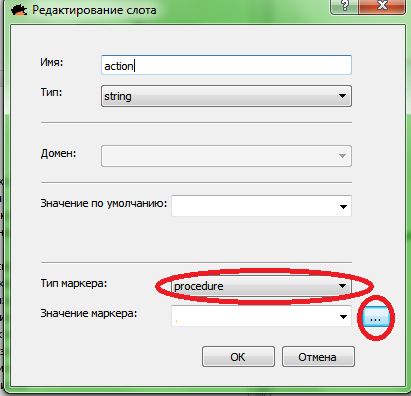
Пишем продукции для означивания слота.



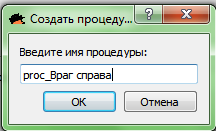
Далее редактируем слот #action для задания поведения игрового объекта.



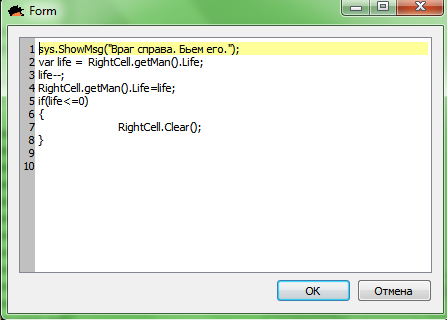
Создаем процедуру.



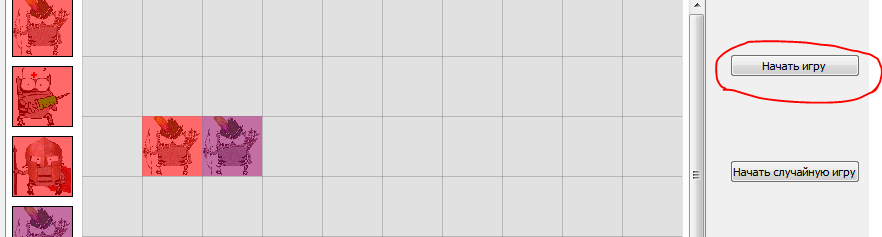
Вводим имя процедуры.



Пишем процедуру на скриптовом языке QScript. Если справа враг, то отнимаем у него единицу жизни. А если жизнь кончилась, то очищаем клетку с врагом.

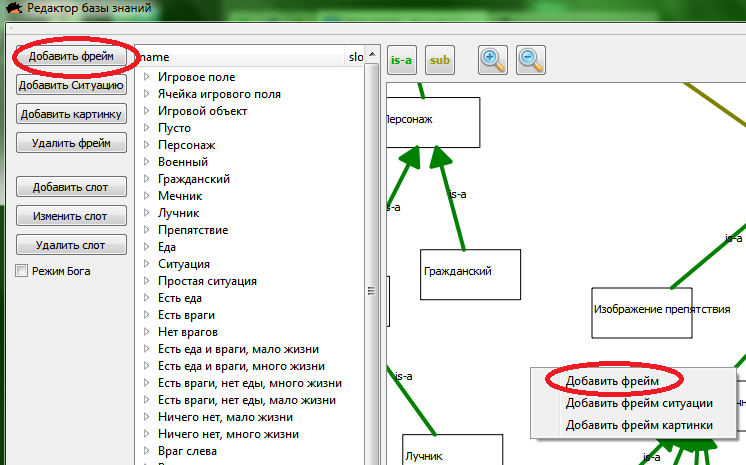


Готово! Ситуация добавлена. Можно тестировать.

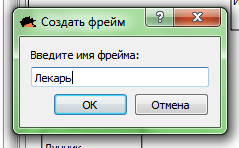


#### Как добавить персонажа

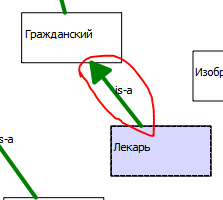
Добавляем фрейм персонажа.



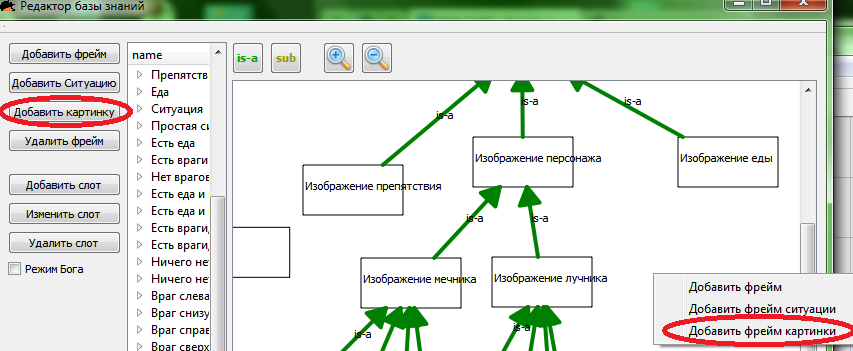
Задаем имя персонажа.



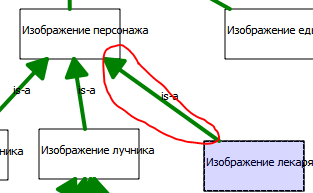
Протягиваем связь is\_a к фрейму “Персонаж” или к его потомку.



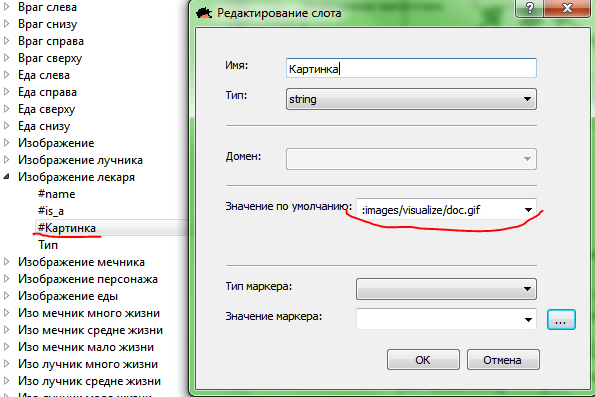
Добавляем фрейм картинки персонажа.



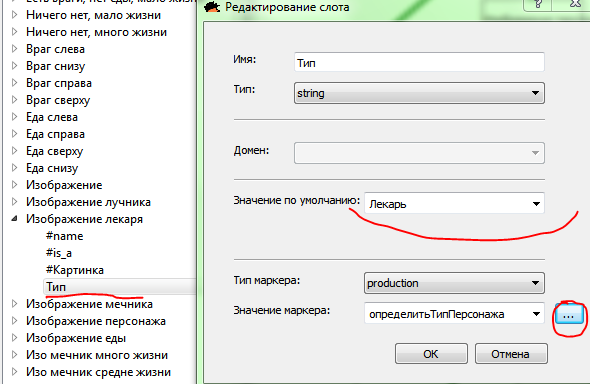
Создаем связь is\_a с фреймом “Изображение персонажа”.



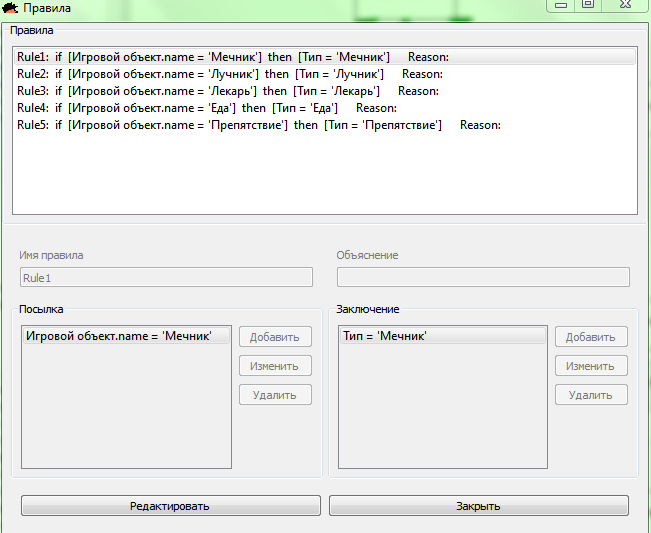
Задаем путь к картинке отредактировав слот Картинка в фрейме Изображение лекаря.



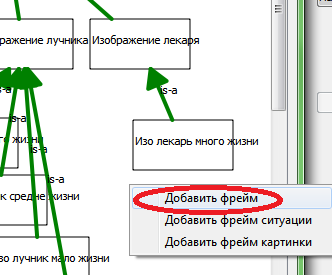
Задаем тип персонажа. Редактируем слот “Тип”.



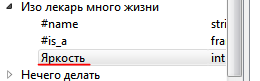
Пишем продукции для определения типа персонажа.

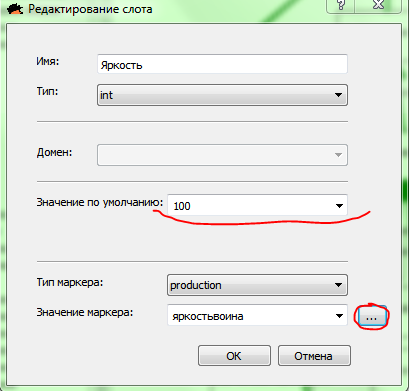


Добавляем фрейм “Изо лекаря много жизни”.

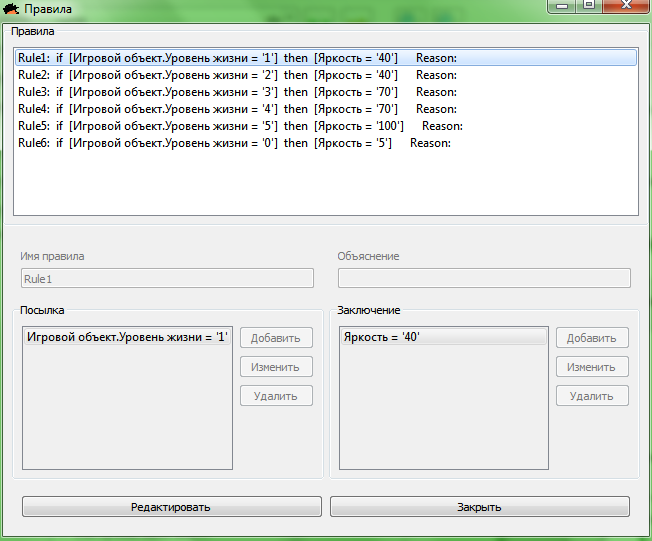


Добавляем слот “Яркость”.





Пишем продукции для привязки яркости к уровню жизни.



Аналогично добавляем фреймы для других значений яркости.



Готово! Лекарь стал доступен для игры.



#### Для чего нужны продукции

Продукции применяются в продукционных слотах. Они используются для анализа содержимого других слотов и установки значений.

Например: Фрейм Есть еда. Слот: рядом еда. Устанавливается значение Да если слева еда, или справа еда, или снизу еда, или сверху еда.

#### Для чего нужны процедуры

Процедуры нужны для совершения действий игровыми объектами. В каждой ситуации есть системный слот #action, который является процедурным. При помощи процедур задается поведение(перемещение, борьба, поглощение пищи).

#### Описание языка процедур

Язык семейства ECMA-script, т.е. по синтаксису сходен в JavaScript.

Типы:

**TObject**  - базовый объект (фрейм).

* getSlotValue(String slotName) : Variant; - получает значение слота
* setSlotValue(String slotName,Variant value) : void; - устанавливает значение слота.

**TCell** : TObject - ячейка игрового поля.

* X: int - свойство координата X
* Y: int - свойство координата Y
* Clear(): void - очищает ячейку
* hasMan(): bool - есть ли человек
* getMan(): TMan - получает человечка из клетки
* getGameObject(): TGameObject - получает игровой объект из клетки
* hasGameObject(QString objName):bool - есть ли игровой объект в клетки с таким именем.

**TMan** : TObject

* Life :int - свойство получает и задает жизни у персонажа.

**TGameObject** : TObject

Скрипту доступны следующие объекты:

Cell : TCell - ячейка игрока.

LeftCell : TCell - ячейка слева

RightCell : TCell - ячека справа

TopCell : TCell - ячейка сверху

BottomCell : TCell - ячейка снизу.

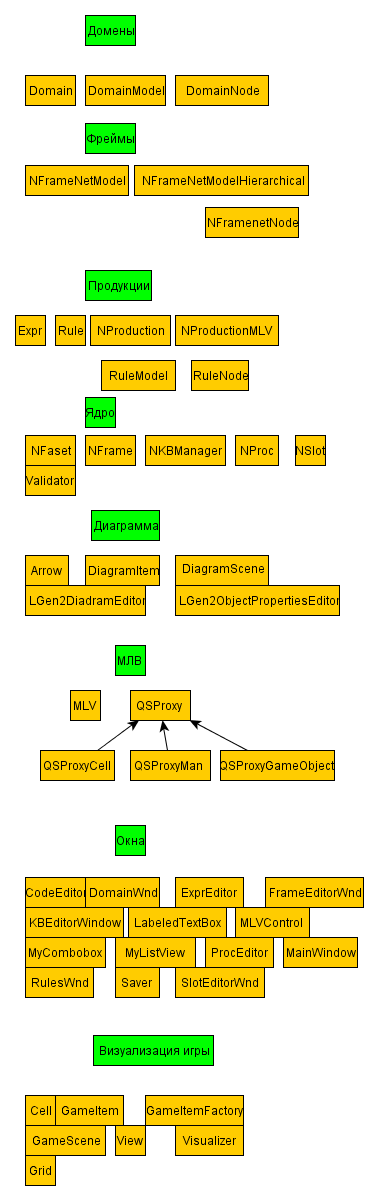
sys.ShowMsg(QString msg); - напечатать сообщение.

sys.NowhereToGo(Cell); - вызвать ситуацию некуда идти.

sys.rand(int maxrand); - сгенерировать случайное число от 0 до maxrand.

## Инструкция разработчика

#### Диаграмма классов



#### Описание классов

Domain - представление домена.

DomainModel - модель доменов

DomainNode - вершина в модели доменов.

NFramenetModel - модель фреймов.

NFramenetModelHierarchical - иерархическая модель фреймов для КО.

NFramenetNode - вершина в модели фреймов.

Expr - выражение в посылке или заключении правила.

Rule - продукционное правило

NProduction - продкуция.

NProductionMLV - механизм продукционного логического вывода.

RuleModel - модель правил.

RuleNode - вершина в модели правил.

NFaset - представление фасета.

NFrame - фрейм.

NKBManager - мереджер БЗ.

NProc - процедура.

NSlot - слот.

Validator - класс для вылидации БЗ.

Arrow - класс стрелки.

DiagramItem - объект диаграммы.

DiagramScene - сцена на диаграмме.

LGen2DiagramEditor - контролл диаграмма.

LGen2ObjectPropertiesEditor - редактор свойств.

MLV - механизм логического вывода по сети фреймов.

QSProxy - базовый класс для объектов, доступных из скриптов.

QSProxyMan - обертка для персонажа для доступа из скриптов.

QSProxyCell - обертка для ячейки игрового поля для доступа из скриптов.

QSProxyGameObject - обертка для игрового объекта для доступа из скриптов.

CodeEditor - контролл редактора кода процедур на QScript.

DomainWnd - окно редактирования доменов.

ExprEditor - окно редактирования выражения в посылке или заключении правила.

FrameEditorWnd - окно редактора фреймов.

KBEditorWindow - окно редактора БЗ.

LabeledTextbox - контролл текстбокса с подписью.

MLVControl - окно компоненты объяснения.

ProcEditor - Окно редактора процедур

MainWindow - главное окно с полем.

RulesWnd - окно правил.

Saver - менеджер сохранения.

SlotEditorWnd - окно редактора слота.

Cell - ячейка на поле.

GameItem - игровой объект на поле.

GameItemFactory - фабрика игровых объектов на поле.

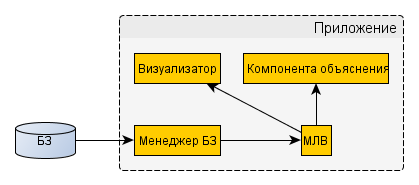
GameScene - игровая сцена.

View - представление.

Visualizer - визуализатор. Отвечает за процесс отображения поля и его содержимого.

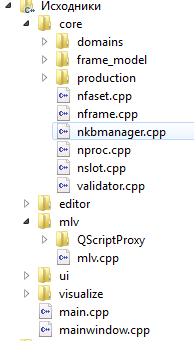
Grid - сетка игрового поля.

#### Архитектура приложения



База знаний хранится отдельно в формате XML. Загружается менеджером БЗ, который управляет БЗ. Когда начинается игра, МЛВ получает из Менеджера БЗ необходимые данные и производит логический вывод. Результаты логического вывода отображаются на поле и в компоненте объяснения.

#### Описание структуры каталогов



**core/** - здесь располагаются классы, которые реализуют основную логику организации фреймовой БЗ.

**editor/** - классы для визуализации БЗ.

**mlv/** - механизм логического вывода.

**ui/** - классы окон.

**visualize/** - классы игрового поля.

**docs/** - сигнатуры функций и классы.

#### Описание МЛВ

Используется наследование same, поэтому вывод идет сверху-вниз.

Для разнообразия поведения используется нетерпеливая стратегия(поиск первой привязавшейся ситуации) и случайный обход фреймов ситуаций.

МЛВ однопроходовый. Сразу после привязки ситуации выполняется действие. Таким образом решена проблема возникновения конфликтов. Картинка персонажа также определяется благодаря привязке.

Если не привязалась ни одна ситуация, то нечего делать - значит привязывается исключительлная ситуация нечего делать.

Если игровой объект окружен и некуда идти, то привязывается ситуация некуда идти. Это выполняется в процедурах.

#### Описание БЗ

База знаний состоит из 2 частей:

● Доменов

● Сети фреймов

В базе знаний присутствуют системные домены и фреймы, которые не могут быть удалены, так как это приведет к полной неработоспособности приложения.

В сети фреймов несколько типов фреймов:

● Игровые объекты

● Игровые ситуации

● Изображения объектов

У каждого фрейма есть по крайней мере 2 системных слота:

#name - имя фрейма

#is\_a - родитель фрейма.

У ситуаций дополнительно:

#action - действие, которое выполняется после привязки ситуации.

У изображений дополнительно:

#Картинка

#Яркость фона

#### Использованные технологии

Qt4 - кросплатформенная библиотека фреймворк для оконного интерфейса.

QtCreator - кросплатформенная IDE для C++.

XML - для хранения базы знаний. (Собственный формат esxml)

QScript - процедуры.

SVN - для управления версиями и совместной разработки. (TortoiseSVN)

Google Docs - для документации.

Doxygen - автоматическая генерация документации по коду.

Разработчики:

* **Бобров Александр Алексеевич.** Ядро, Интерфейс пользователя, Хелп, Процедуры, разработка БЗ,продукции, МЛВ для продукций, тестирование.
* **Киселев Александр Юрьевич.** Игровое поле и визуализация, МЛВ, компонента объяснения, разработка БЗ. тестирование.
* **Сторожев Антон Александрович.** В 3 семестре не участвовал в разработке :-(. Редактор фреймов, разработка БЗ, тестирование.

2012-2013г