Cartoon Online

Krokoz™ | [Адрес организации]

Дизайн документ

SFedulenko

2015

Содержание

[1.Описание игры 2](#_Toc465420031)

[2.Описание режимов 3](#_Toc465420032)

[2.1 Спринт 3](#_Toc465420033)

[2.2 Гонка за ресурсы 4](#_Toc465420034)

[3.Прототип бета-версии 5](#_Toc465420035)

[3.1.Логика приложения 6](#_Toc465420036)

[3.2 Алгоритмы приложения 7](#_Toc465420037)

[3.3 Элементы приложения 10](#_Toc465420038)

[3.4.Архитектура приложения 11](#_Toc465420039)

[3.6.Разработка ИИ 20](#_Toc465420040)

# 1.Описание игры

Игра представляет собой поле. Поле может иметь различную форму, в стандартном наборе - это линейная, круговая, сетка.

Игрока с ресурсами-энергия, здоровье, деньги. Поле поделено на сектора, каждый сектор — это событие.

В игре 7 типов событий: белый, синий, зеленый, желтый, красный, черный, серый.

Белые квадраты предполагают благоприятные события, которые приносят ресурсы. Это может быть работа, божественное вмешательство, случайная находка и т.д.

Синий - испытания, за которые нужно платить энергией.

Зеленый – это магазин можно что-нибудь купить.

Желтый – происходит событие и вариант действия в зависимости от выбора игрока и его текущих параметров произойдёт цепь событий, раскрывающаяся в последующих попаданиях на желтый квадрат.

Красный – испытания, за которые нужно платить здоровьем.

Черный – испытания, за которые нужно платить всеми ресурсами (определенным количеством)

Серый – испытания, за которые нужно платить энергией и деньгами (определенным количеством)

Основная задача игрока будет зависеть от режима:

* Прийти к финишу (последнему сектору) первым.
* Набрать первым определённое количество ресурсов.
* Просто выжить.
* Найти предмет, выполняя квесты и попутно решая проблемы, которые попадаются на пути в виде штрафных секторов.

# 2.Описание режимов

### 2.1 Спринт

Все игроки стартуют с одной позиции. Цель прийти к финишу первым. Игровое поле представляет собой линию. Квестовая линия тоже является прямолинейной, выбор вариантов действий зависит от того, какой ресурс хочет получить игрок. Некоторые квесты предназначены для помощи прохождения, некоторые чтобы отдалить игрока от победы.

Игрок перемещается по полю главным образом за счет очков на текущем броске игральной кости. Единица дает бонус к энергии в размере 10%, шестерка дает 10% к здоровью.

Игрок может перемещаться при попадании на некоторые клетки, при выборе варианта задания. Если игрок перемещается на клетку путем отправления, события, связанные с этой клеткой он игнорирует.

В режиме спринт отсутствуют белые квадраты.

В случае полпадения на одну клетку с другим игроком. У них происходит битва, в которой решает энергия, у игрока с наименьшим количеством энергии отнимается здоровье в величине равной разнице между их энергиями. Так же урон оппоненту можно нанести с использованием некоторых товаров в магазине, в течении выполнении квеста. Информация о игроках скрыта друг от друга. Ее можно узнать только, оказавшись на клетке, занимаемой игроком, или оказавшись в шаге одной клетки, или через товар в магазине.

У игрока есть статус:

* Карма – определяет увеличение/понижения бонусов и штрафов. Пропуск ходов. Варианты бросков кубика. И т.д.
* Расположение на предыдущей клетке
* Расположение на текущей клетке
* Расположение на клетке после броска кости
* Очки здоровья
* Очки энергии
* Количество денег
* Желтый – определяет текущий квест и цепочку задания.

### 2.2 Гонка за ресурсы

Игровое поле представляет собой круги с секторами. При достижении определенных условий игрок поднимается на следующий уровень где секторов больше, больше ресурсов и больше проблем.

# 3.Прототип бета-версии

Прототип бета версии, платформа Windows, консольный вывод, режим спринт.

Задачи:

1. Описать логику приложения
2. Описать алгоритмы приложения, алгоритм игрового процесса, запуска/остановки игры, ведение журнала данных, сохранение текущих данных игры
3. Разработать элементы, классы, библиотеки приложения
4. Проект архитектуры приложения(общая схема классов)
5. Описать все члены, функции, методы каждого класса
6. Разработать ИИ

## 3.1.Логика приложения

Инициализация стартового меню, инициализация необходимых классов.

Настройка опций

Выбор типа игры

Подготовка доски

Подключение игроков

Игровой процесс

Сохранение

## 3.2 Алгоритмы приложения

1. Инициализация стартового меню, инициализация необходимых классов.
2. Настройка опций
3. Выбор типа игры
4. Вызов менеджера игры

0

1

Создать игру

Ввод параметров

Ввод пользователя

Ввод пользователя

Ввод параметров

Загрузить игру

Опции

Выход

Новая игра

Создать меню

Показать заставку

5.Менеджер игры

Запуск всех модулей игры

1

Создать игроков

Создать игровое поле

Подключить игроков

Действие игрока

Игровое поле

N+1

Игрок N

2

Правила

Игровой процесс

Ход игрока N завершен

нет да

0

Игрок N в игре

нет

да

Состояние игрока

6.Загрузка/сохранение

Ход игрока N завершен

2

нет да

Сохранить состояние игрока в файле

Сохранить состояние игрока в памяти

Состояние игрока

1

## 3.3 Элементы приложения

AppManager (запуск приложения, вызов подсистем)

AppInterface(пространство имен содержащий элементы управления, меню, графику)

InterfaceType(класс определяющий тип интерфейса)

BaseInterface(основной класс интерфейсов)

ConsoleInterface(подкласс интерфейса консольного приложения)

WindowInterface(подкласс интерфейса оконного приложения)

SVmanager(класс для хранения и загрузки состояний игры)

SVData(данные об игре)

COption (класс для хранения настроек игры)

GameType (класс определяющий тип игры для дальнейшего вызова соответствующего менеджера игры)

SprintMode(подсистема игрового процесса)

TreasureHuntMode(подсистема игрового процесса)

GameManager(вызывает подситемы для игрового процесса)

PlayerType(класс типов игроков)

Player(класс игрока)

PlayerPool(класс для управления игроками)

PlayerTurn(класс реализующий ход игрока и его последствия)

BoardManager(модуль для управления доской, окружающим миром и т.д)

BoardTemplate(базовый класс игровой доски)

SprintBoard(доска для режима спринт)

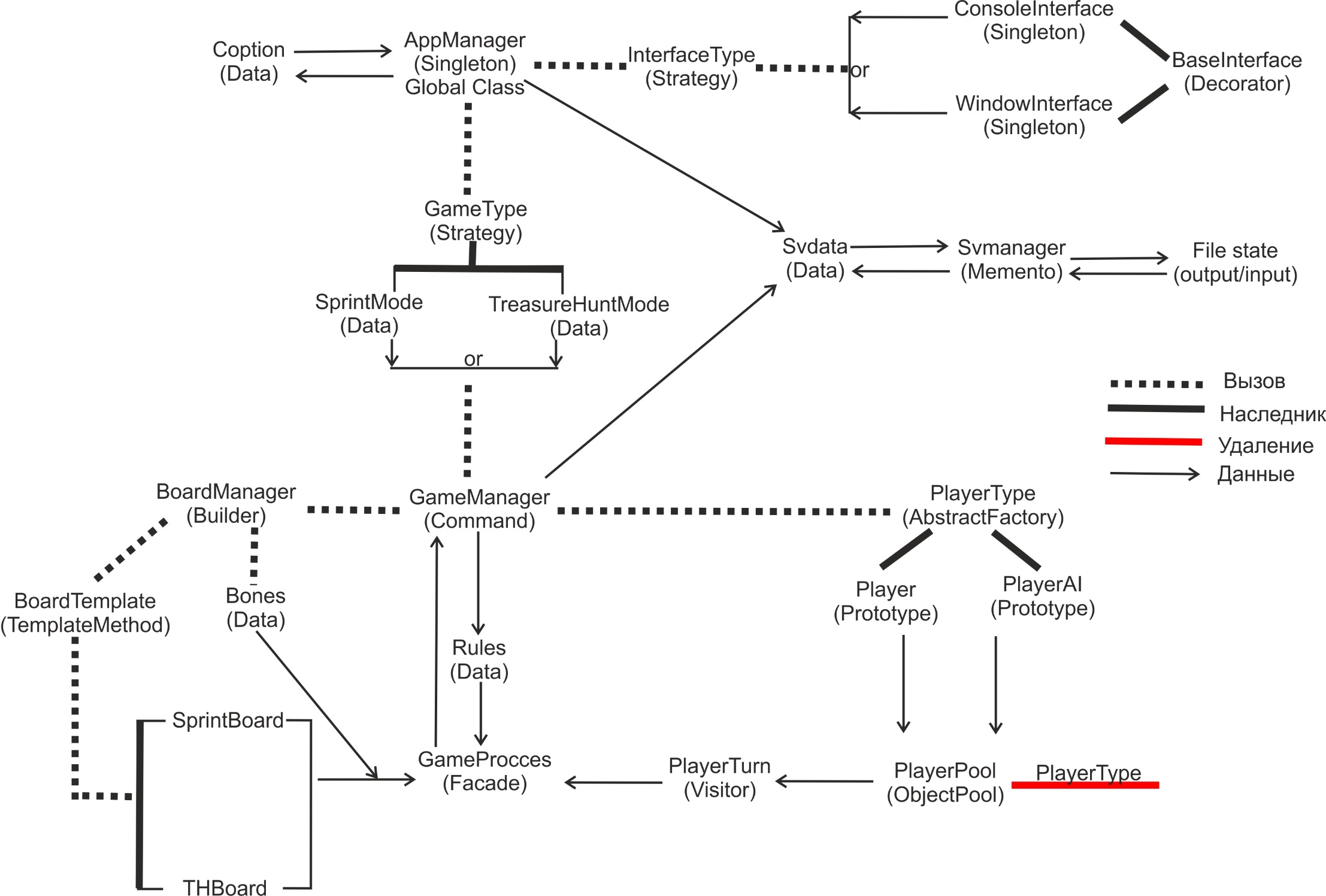
THBoard(доска для режима гонка за ресурсы)

Bones(класс для методов перемещения игроков)

Rules (описание правил и условий для игрового процесса)

GameProcess(управление подсистемами игры)

## 3.4.Архитектура приложения



3.5.Члены классов

Таблица AppManager

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| Subsystemtype | subSystemOptionGet | subSystemOption value | Получает подсистему для компоновщика |
| void | subSystemOptionSet | subSystemOption value | Задает подсистему для компоновщика |
| void | ShowTitle | Title(AppInterface) | Показать заставку |
| void | CreateMenu | Menu(AppInterface) | Создать меню |
| char | GetUserInput | Input\_keyboard | Считать данные пользователя |
| void | CreateGame | GameType | Создать игру |
| void | LoadGame | LoadData | Загрузить игру |
| void | CreateOptionMenu | OptionMenu | Создать меню опций |
| void | EnterOptionMenu | OptionMenu | войти в меню |
| void | LeaveOptionMenu | OptionMenu | Выйти из меню |
| Coption | GetUserOptions | Coption | Сохранить в память настройки |
| void | Exit |  | Закрыть приложение |
| void | CreateTypeGameMenu | GameMenu,  gametypedata  data | Создать меню выбора типа игры |
| GameType | GetGameType | User\_input | Сохранить выбранный тип игры |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Таблица BaseInterface(namespace)

|  |  |  |  |
| --- | --- | --- | --- |
| Класс | Функция | Переменные | Примечание |
| Text | AddItem | template | Добавить строку |
| Menu | SetTitle | title | Заголовок меню |
|  | AddItem | items | Добавить элемент меню |
|  |  |  |  |
| IControl(базовый) | DecorateOperation | virtual | Декорация элемента управления |
| DecoratorConsole | Decorate(control) | virtual |  |
| DecoratorWindow | Decorate(control) | virtual |  |
| ConsoleInterface |  |  |  |
| WindowInterface |  |  |  |
| InterfaceType | Switch(DecoratorConsole) | <Controls> | Выбор интерфейса |
| Класс ConsoleInterface | | | |
| Тип значения | Функция | Переменные | Примечание |
| Void | ShowMessage | template | Показать сообщение |
| char | Get\_input\_keyboard |  | Получить значение клавиши клавиатуры |
| void | SetCursor | (x,y) | Установить позицию курсора |
| void | drawrectangle | X,y,x1,y1 | Нарисовать квадрат |
| void | drawline | X,y | Нарисовать линию |
| Void | drawgrid | X,y,x1,y1 | Нарисовать сетку |
| Void | drawarraypicture | array | Нарисовать узор с помощью массива |

Таблица GameType

|  |  |  |  |
| --- | --- | --- | --- |
| Класс | Функция | Переменные | Примечание |
| Game | StartGame | virtual |  |
| SprintGame | StartGame | (sprintmode) | Старт игры спринт |
| THGame | StartGame | (THmode) | Старт игры гонка |

Таблица GameManager

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| void | BoardFactory | (board type) | Запустить строителя доски |
| void | PlayerFactory | (playerorder) | Запустить фабрику игроков |
| Void? | FillPlayerPool | Vector<player> | Включить игроков в пул |
| void | LuanchGameProcess | (board,player,rules) | Запустить игровой процесс |
| SVdata | SaveState | (playerstate,boardstate) | Сохранить параметры |
|  |  |  |  |
|  |  |  |  |

Таблица GameProcess

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| playerstate | CheckPlayerState | (boardpos,playerstate) | Состояние игрока |
| boardstate | GetBoardInfo | (board) | Состояние доски |
| bool | Is\_alive | (rules,curplayer) | Игрок жив? |
| bool | Is\_canmove | (rules,curplayer) | Игрок может совершить ход? |
| bool | Is\_turnover | (playerinput) | Игрок завершил ход? |
| int | NextPlayerTurn | (playerinput,playerspool) | Игрок завершил ход |
| void | BoardEventSingle | (current player,boardevent) | Событие для текущего игрока |
| void | BoardEventAll | (for each playerin pool,boardevent) | Событие для всех игроков |
| void | BoardEventSelected | (for selected players in pool,boardevent) | Событие для выбранных игроков |
| void | SendCurData | (board,boardpos,plaerstate) | Послать данные менеджеру |
| Void | SetLoadedData | (board,boardpos,plaerstate) | установить данные |
| int | GetBoneData | (Bones::Throw) | Получить очки передвижения |

Таблица PlayerType

|  |  |  |  |
| --- | --- | --- | --- |
| Класс | Функция | Переменные | Примечание |
| PlayerType | SetName | (name) | Установить имя |
|  | GetName | (name) | Узнать имя |
|  | CreatePlayer | (id,name) | Создать игрока |
| Player | AddPlayer | (id,name) | Добавить игрока |
|  | RemovePlayer | (id,name) | Убрать Игрока |
|  | GetPlayerInfo | (playerdata) | Получить параметры игрока |
| PlayerPool | AddToPool | (Player) | Добавить в пул |
|  | ReturnPlayer | (Player) | убрать |
|  | GetPlayer | (Player) | Сделать активным |
|  | GetPoolSize |  | Узнать количество игроков |
|  |  |  |  |

Таблица PlayerTurn

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| playerparams | MakeTurn | (bonesThrow) | Кинуть кубики |
| action | CreateActionMenu | (Menu) | Создать меню выбора |
| playerparams | GetPlayerParams | (curr pool player) | Получить парметры |
| id | EndTurn |  | Завершить ход |
|  |  |  |  |

Таблица BoardManager

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| board | GetBoard | (boardparams) | Получить параметры доски |
| void | BuildingBoard | (boardtype) | Запустить строительство доски |
| void | PrepareParts | (partstype,partsinfo) | Создать элементы доски |

Таблица BoardTemplate

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| boardtype | GetBoardType | (user input) |  |
| void | Start | () |  |
| void | End | () |  |
| void | SetupPartsEvents | (mode data) | Установить события доски |
| partinfo | GetPartInfo | (part) | Узнать параметры части |
| virtual | CreatePart | (type,pvalue) | Данные для части |
| virtual | CreateQuest | (quest) | Добавить квест |
| virtual | AddEvent | (event) | Добавить событие |
|  | | | |
| Класс | Функция | Переменные | Примечание |
| SpintBoard | CreatePart | (type,pvalue) | Данные для части |
|  | CreateQuest | (quest) | Добавить квест |
|  | AddEvent | (event) | Добавить событие |
| THBoard | CreatePart | (type,pvalue) | Данные для части |
|  | CreateQuest | (quest) | Добавить квест |
|  | AddEvent | (event) | Добавить событие |

Таблица Bones

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| Int | ThrowBones | (faces) | Бросок кубика |
| int | Modificator | (mod) | Режим/вариации броска кубика |
| int | SetFaceses | (int) | Установить количество граней кубика |

Таблица SVmanager

|  |  |  |  |
| --- | --- | --- | --- |
| Тип значения | Функция | Переменные | Примечание |
| playerstate | SavePlayerState | (playerstate)) | Сохранить |
| boardstate | SaveBoardState | (boardparams,boardtype,partsinfo) | Сохранить |
| Void | Load | (filename) | Загрузить |
| void | Load | (filename) | Загрузить |
|  |  |  |  |

## 3.6.Разработка ИИ