Неформальное заведение JaggedStudio
E-mail: Jagged.N@yandex.ru
Steam: https://steamcommunity.com/id/JaggedNel/

СПРАВОЧНОЕ ПОСОБИЕ ВНУТРИИГРОВОГО ПРОГРАММНОГО КОМПЛЕКСА

ПРЕДВАРИТЕЛЬНАЯ ВЕРСИЯ

Разработчик и исполнитель: JaggedNel
Дата начала реализации проекта: 23.12.2019
Дата, на которую актуально содержание пособия: 03.08.2020
Актуальная версия Nelbrus: 0.3.10 [05.03.2020]
Санкт-Петербург 2019-2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Использование внутриигровых программ в игре Space Engineers часто становится решением прикладных задач всевозможного спектра при реализации различных проектов любых уровней. Однако, системы могут использовать неудобные или неэффективные методы настройки и работы. При этом в отдельно взятом проекте может использоваться множество программ, которые, при необходимости, трудоёмко укомплектовать в один программируемый блок.

В связи с этим, получение инструмента эффективного для разработки и взаимодействия с прикладными системами представляется одной из перспективнейших задач. Система Nelbrus призвана стать таким инструментом.

Данный свод документации создан с целью подробно ознакомить пользователей комплекса с его функционалом, помочь при разработке дополнительных компонентов и комплексов для системы. Пособие подразумевает наличие у его пользователя наличие основных знаний о внутриигровых скриптах. Если их нет, рекомендуется ознакомится с внутриигровым программированием перед началом ознакомления.

О любых ошибках и недочётах содержания или работы системы прошу сообщать через соответствующие обсуждения на страницах работы в Workshop[2] или репозитории[3].

ОПРЕДЕЛЕНИЯ

Компонент – набор команд, рассматриваемый как единое целое, выполняющий законченную функцию и применяемый самостоятельно или в составе комплекса.

Комплекс – набор команд, состоящий из двух или более компонентов или комплексов, выполняющих взаимосвязанные функции, и применяемый самостоятельно или в составе другого комплекса.

Ядро – комплекс системы Nelbrus, не включающий сторонние дополнения, подпрограммы и т.п.

Материнский блок – программируемый блок с системой Nelbrus.

Программа – полный перечень компьютерных инструкций и данных, записанных в программном блоке.

Подпрограмма – программный комплекс или компонент, под управлением Nelbrus в составе программы. В программе родителем класса подпрограммы является класс *SubP*.

Сборка – программа, содержащая ядро вместе с любым набором компонентов, комплексов или подпрограмм.

Тик – минимальная внутриигровая единица времени, равная 1/60 реальной секунды при скорости симуляции 1,00.

1 Описание программного комплекса

Систему Nelbrus следует рассматривать как операционную систему, созданную для использования в среде внутриигровых скриптов игры Space Engineers, выполняющую следующие основные функции:

1. Организация среды для функционирования прикладных подпрограмм;

1.1. Количество различных подпрограмм, используемых в одной программе ограничено длинной программы выраженной в символах (100 000 символов по умолчанию).

1.2. Количество подпрограмм, запущенных единовременно программно не ограничено.

2. Обеспечение доступа пользователя к ресурсам и функционалу системы и подпрограмм посредством:

2.1. Базового командного интерфейса взаимодействия Nelbrus, предлагаемого к поддержке подпрограммами любых типов и назначений.

2.2. Другими типами интерфейсов, встроенных в подпрограммы или добавленные в программу иными способами.

Цель разработки системы – получение унифицированных методов разработки и управления внутриигровыми программными проектами.

1.1 Структура программного кода системы

На рисунке 1 приведена схема общего устройства сборки системы.

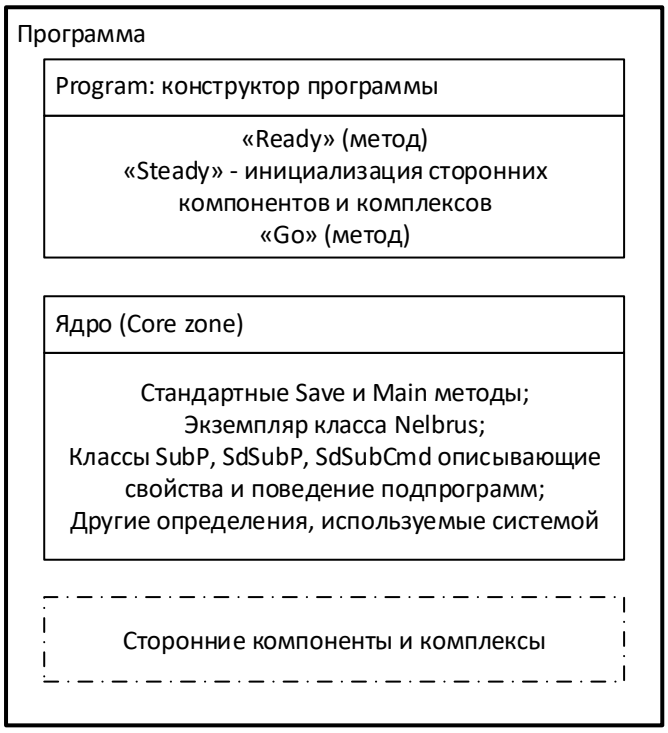

Рисунок 1 – Структура сборки

Программа состоит из следующих блоков:

- Стандартного конструктора *Program*, вызываемого при компиляции программы. Инструкции выполняемые здесь разделены на выполняющиеся один раз после компиляции программы 3 этапа подготовки «Ready, Steady, Go»:

- *Ready* производит первоначальную настройку ОС без чего невозможна выполнения других этапов. Представляет собой вызываемый метод.

- *Steady* инициализирует сторонние компоненты и комплексы (подпрограммы) в области видимости ОС. Является группой методов (SetEchoCtrl, ISP).

- *Go* завершает настройку ОС и переводит её в рабочий режим. Является методом.

- Ядра системы, содержащего: стандартные методы скрипта, родительские классы подпрограмм и другие компоненты системы.

- Определений сторонних подпрограмм, комплексов и компонентов. При этом они должны вноситься в область видимости ОС на этапе Steady в конструкторе программы.

1.2 Структура ядра системы

На рисунке 2 приведены комплексы и компоненты, составляющие ядро. Стрелками указано направление наследования классов.

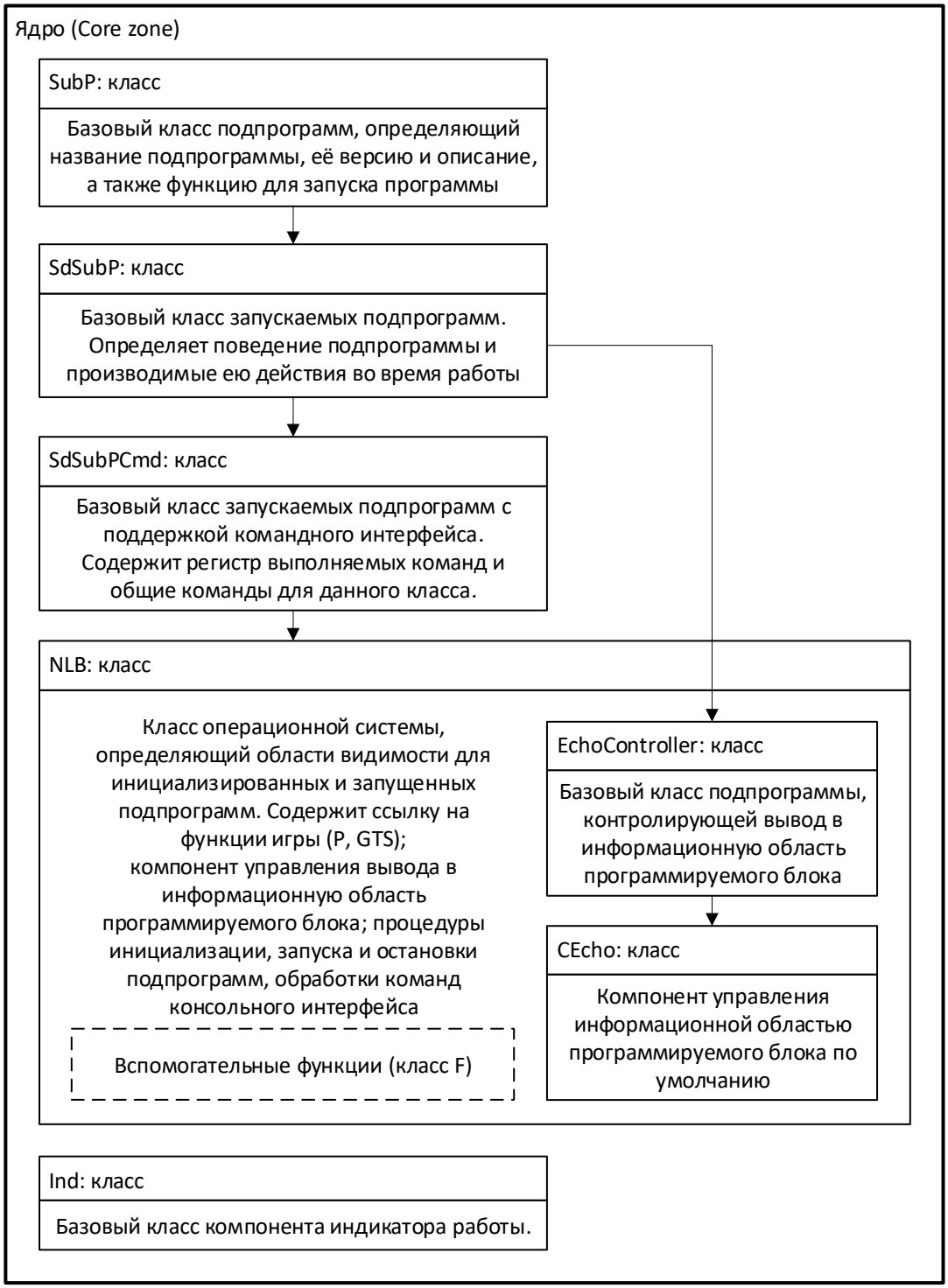

Рисунок 2 – Архитектура ядра системы Nelbrus

2 Установка и обновление

Есть два основных источника получения программных компонентов и комплексов:

1. Steam Workshop;
2. Репозитории.

Система Nelbrus доступна через оба источника:

- через подписку на сборку нужной версии в Steam Workshop от JaggedNel [4] и последующую частичную или полную вставку кода программы в программируемый блок;

- частичное или полное копирование кода программы из репозитория JaggedNel [3], где хранятся доступные версии комплекса и документация.

Всегда перед установкой любого программного обеспечения тщательно ознакомьтесь с рекомендациями автора по установке или обновлению. Ниже приведены общие инструкции по установке и обновлению.

2.1 Чистая установка ядра

Для любых целей возможна чистая установка ядра системы наиболее подходящим способом с полным копированием кода сборки в программируемый блок. После этого комплекс будет доступен для использования.

2.2 Обновление ядра

Для обновления ядра программы необходимо полностью заменить код устаревшего ядра системы на новое. Ядро выделено в коде ключевыми фразами: «`#region Core zone`» перед ним и «`#endregion Core zone`» после.

2.3 Установка подпрограмм

Для установки подпрограммы её код добавляется в конце кода программы. В зоне *Steady* конструктора *Program* класс подпрограммы обязательно должен инициализироваться для введения в зону видимости ОС. Инициализация производится методом ISP из ядра ОС следующим образом:

```
OS.ISP(new …());
```

3 Работа комплекса

3.1 Командный интерфейс

Командный интерфейс (КИ/CI) производит выполнение команды системы и команд подпрограмм из реестра подпрограмм с поддержкой командного интерфейса. Ввод команд производится через:

- Обработку аргумента, с которым вызывается материнский блок.

Строка, содержащая команду, должна начинаться с символа начала команды (/). Затем название команды, которая будет выполнена, и, при необходимости, через пробел перечислены аргументы, предусмотренные синтаксисом команды. Таким образом общий вид команд имеет вид:

/command_name

/command_mame argument1 argument2 … argumentN

В случае если один из аргументов имеет в своём значении пробелы, то во избежание его разделения, перед ним и после него следует добавить символ скрепления - одинарную кавычку (’). Кроме того, последний аргумент, начавшийся символом скрепления, но не закончившийся им будет считаться скреплённым. Например:

/command_name ’single argument’

/command_name argument ’single argument

Каждая подпрограмма с поддержкой консольных команд имеет базовую команду получения помощи *help*. Поддерживаемый синтаксис команды *help*:

/help – вызов команды без аргументов показывает список доступных команд в реестре;

/help command_name – вызов команды с аргументом-именем команды из реестра для получения подробной информации о ней.

Все вводимые команды выполняются из реестра операционной системы. Для выполнения команды из реестра команд другой подпрограммы следует пользоваться командой *sp*.

Перечень доступных команд системы приведён в конце пособия.

3.2 Периодические и отложенные действия подпрограмм

Каждая запущенная подпрограмма, определяемая классом «SdSubP», может выполнять различные действия с заданной частотой. За хранение отвечает делегат (void Act) «EAct» (Every tick actions) (выделен отдельно с целью оптимизации) и коллекция делегатов «Acts».

Ключом коллекции «Acts» (`Dictionary<uint, Dictionary<uint, Act>>` = <тик, < частота, делегат>>) является тик, в который должны будут быть выполнены делегаты. Значением коллекции «Acts» является коллекция с ключом-частотой.

Для управления периодическими действиями в подпрограмме определяется переменная типа *CAct* (Custom Action) и используются следующие функции:

- Создание: *AddAct*
- Удаление: *RemAct*
- Изменение: *ChaAct*

Также доступно использование отложенных действий управляемых методами:

- Создание: *AddDefA*
- Удаление: *RemDefA*

ПЕРЕЧЕНЬ ДОСТУПНЫХ КОМАНД СИСТЕМЫ

Аргументы команд, обрамлённые символами <…> являются обязательными. Аргументы, обрамлённые символами […] являются дополнительными

Команда	Описание	Детали
start	Запускает инициализированную подпрограмму по id.	Пример: */start <id>* Проверка id производится по команде «/isp».
stop	Останавливает запущенную подпрограмму по id.	Пример: */stop <id>* Проверка id производится по команде «/sp».
sp	Просмотр запущенных подпрограмм или выполнение команды подпрограммой.	Пример: */sp* – (Без аргументов) Просмотреть список запущенных подпрограмм; */sp <id>* – Просмотреть информацию о запущенной подпрограмме по *id*; */sp <id> <command> [arguments]* – Выполнение подпрограммой с *id* команды *command* и аргументами *arguments*.
isp	Просмотр инициализированных подпрограмм.	Пример: */isp* – (Без аргументов) Просмотр списка инициализированных подпрограмм; */isp <id>* - Просмотреть информацию об инициализированной подпрограмме по *id*.
clr	Очистить командный интерфейс.	Пример: */clr*

ПОЛЕЗНЫЕ ССЫЛКИ

1. Steam-профиль JaggedNel: https://steamcommunity.com/id/JaggedNel/
2. Актуальная версия Nelbrus в Steam Workshop: https://steamcommunity.com/sharedfiles/filedetails/?id=2014553432
3. Репозиторий Nelbrus: https://github.com/JaggedNel/Nelbrus
4. Мастерская Steam JaggedNel: https://steamcommunity.com/id/JaggedNel/myworkshopfiles/?appid=244850