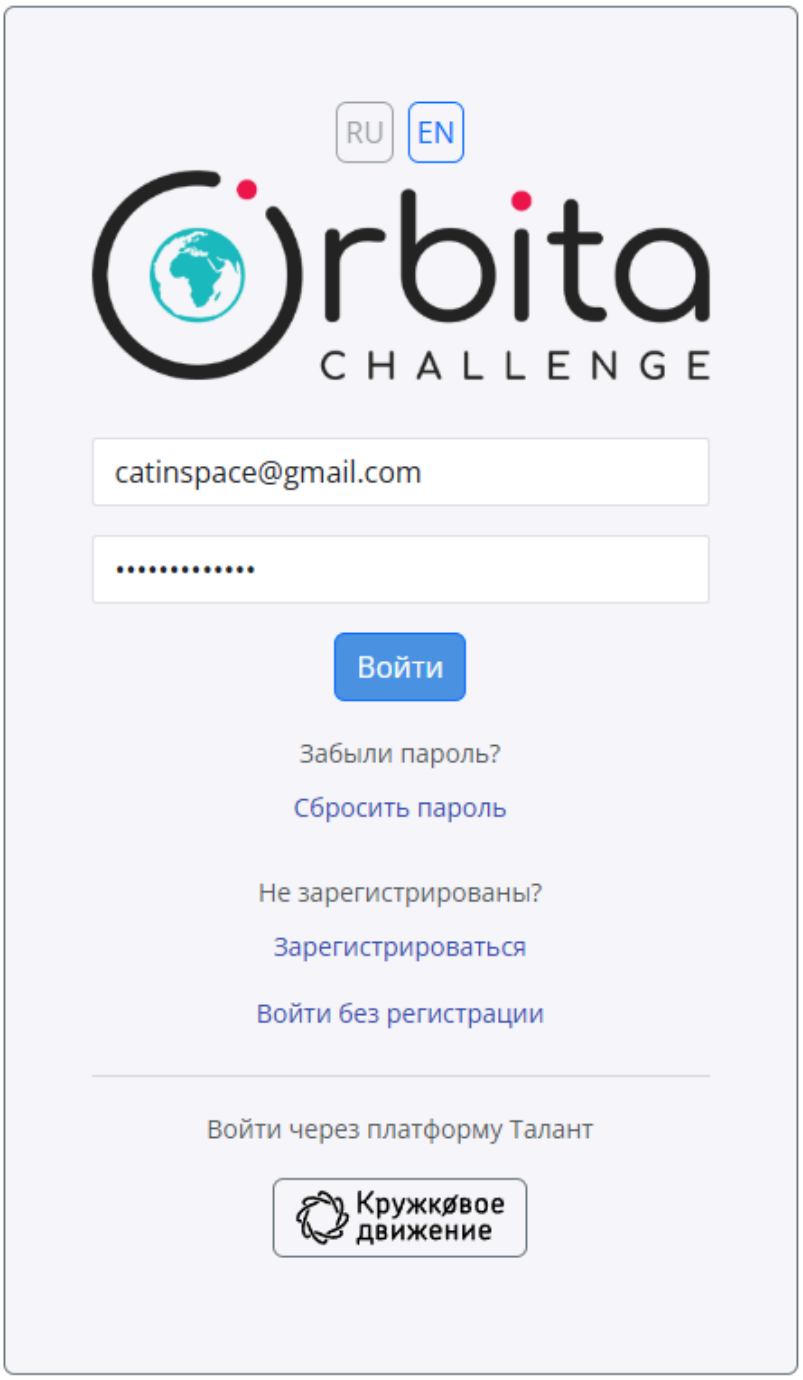
ООО "Образование будущего"

РУКОВОДСТВО АДМИНИСТРАТОРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ «ОРБИТА.ЧЕЛЛЕНДЖ»

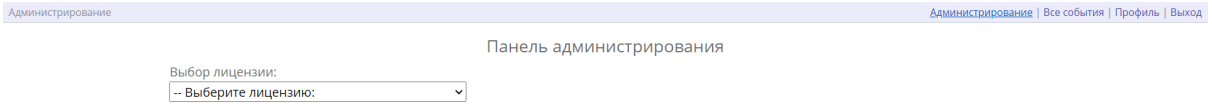
г. Москва
2024 г.

1. Начало работы

Вход для всех пользователей осуществляется по ссылке: https://orbita.education/



После входа в систему пользователь попадает на страницу с доступными ему событиями. Чтобы попасть на панель администрирования, необходимо кликнуть на “Администрирование” в правом верхнем углу сайта.



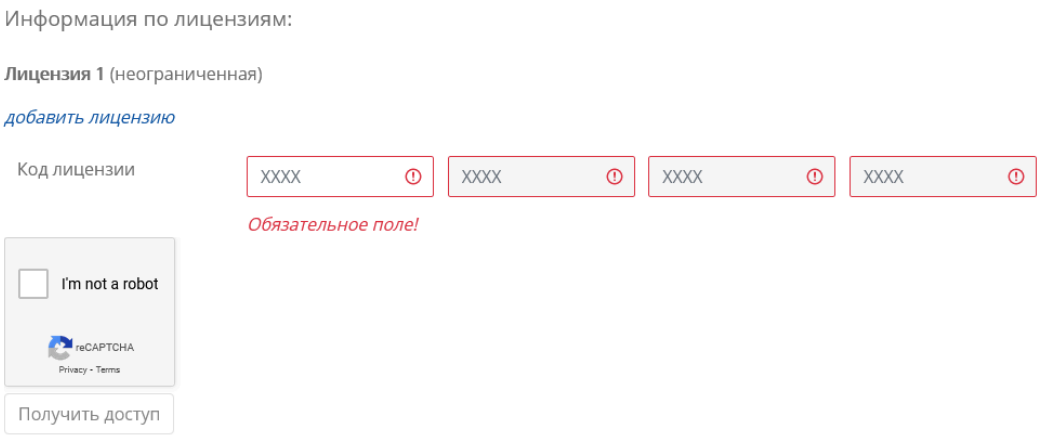
На панели администрирования требуется выбрать подходящую лицензию, тогда станет доступным выбор функций, которые позволяют:

- управлять событиями и сценариями;
- просматривать участников;
- изменять группы, события и команды;
- добавлять изображения, которые могут быть использованы в событиях и их элементах;
- просматривать решения, отправленные участниками;
- корректировать оценки для команд и участников;
- загружать списки участников для массовой регистрации;
- просматривать каталог с типовыми сценариями, страницами и курсами, которые могут быть добавлены в события (затем, при необходимости, их можно изменять).

Если на панели администрирования нет доступных лицензий, при наличии секретного кода можно активировать лицензию самостоятельно (см. раздел “Активация лицензии”).

2. Активация лицензии

Для получения статуса администратора лицензии зарегистрированному пользователю необходимо кликнуть на “Профиль” в правом верхнем углу сайта, затем найти строку “добавить лицензию”, ввести в появившееся поле код лицензии, поставить галочку в поле “I’m not a robot” и кликнуть на “Получить доступ”.



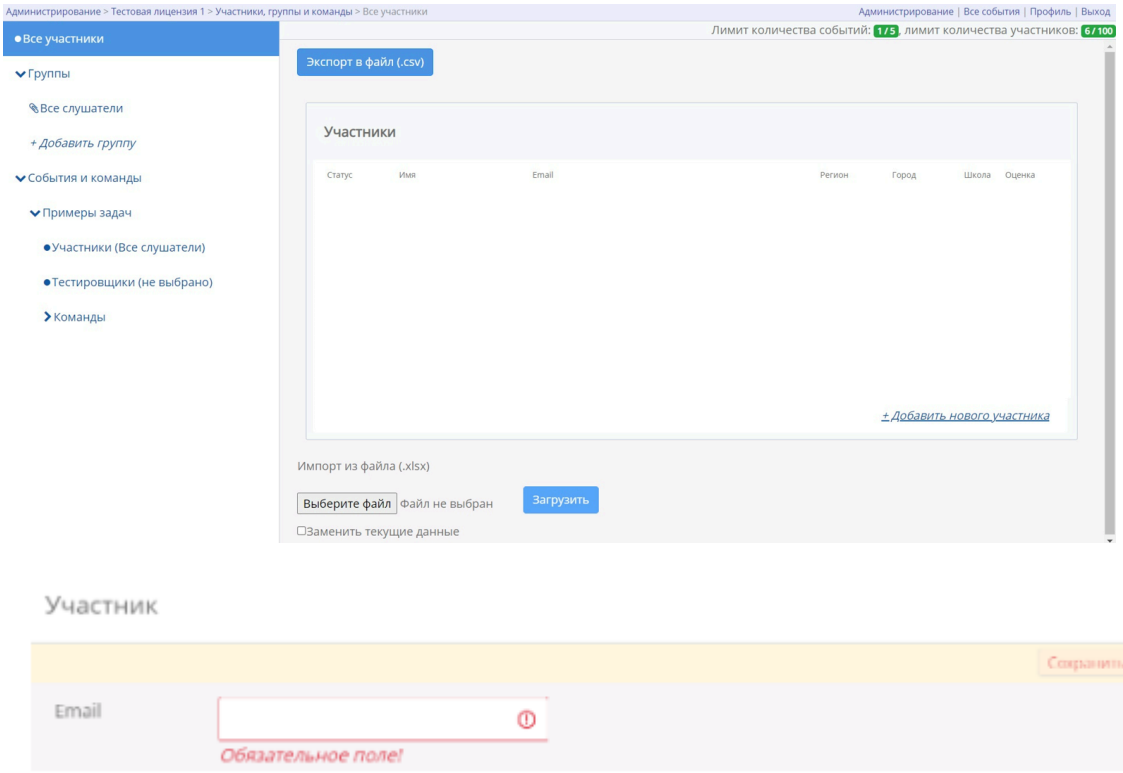
3. Управление участниками

Внимание: уникальным идентификатором участника в системе является его адрес электронной почты. В случае добавлению к событию незарегистрированного участника, он получит доступ к событию после регистрации в системе по адресу https://nti.orbitagame.ru/.

3.1. Добавление участников и удаление участников

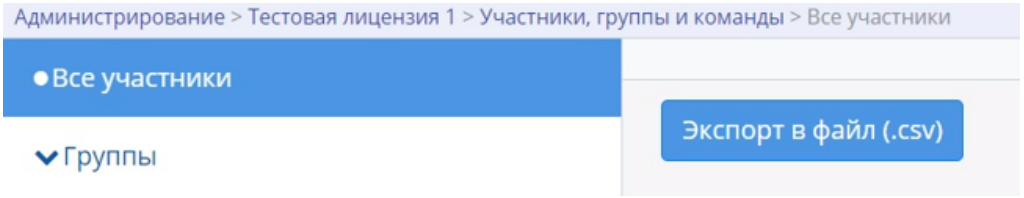
Чтобы добавлять новых участников, на панели администрирования нужно выбрать “Участники, группы и команды”. Вверху справа можно увидеть лимиты количества событий и участников, заданные свойствами лицензии.

Во вкладке “Все участники” можно добавить зарегистрированных пользователей, кликнув на “Добавить нового участника” Во всплывающем окне необходимо вписать email-адрес участника.

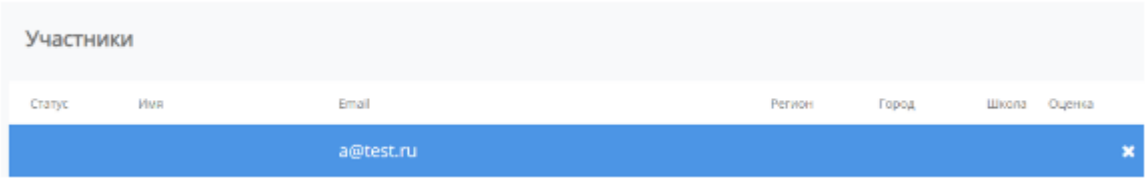


Также возможно добавить сразу несколько email-адресов, сохранив их в таблицу в формате .xlsx и загрузив эту таблицу на сайт.

При необходимости можно скачать сформированный список участников в формате .csv.

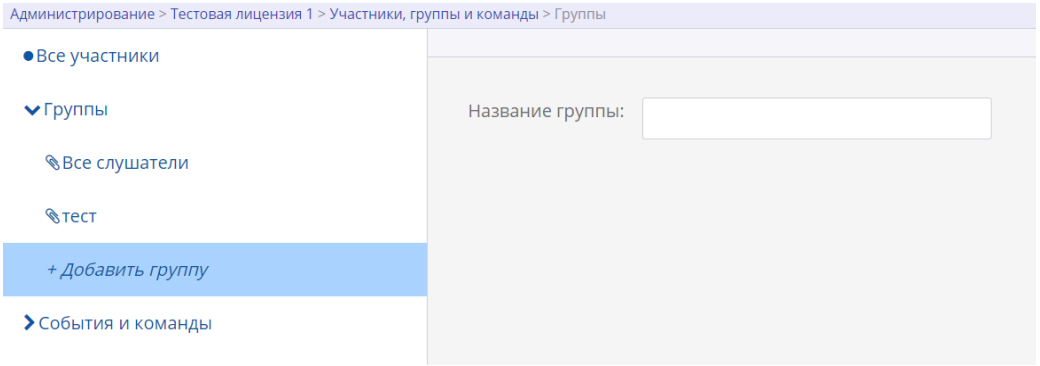


Для удаления участника требуется нажать на крестик, который всплывает справа строки при наведении курсора мыши на интересующего участника.

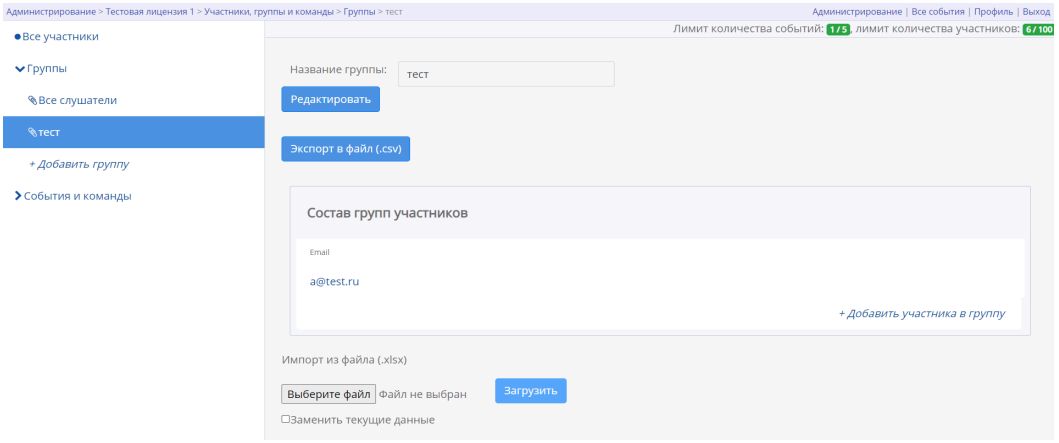


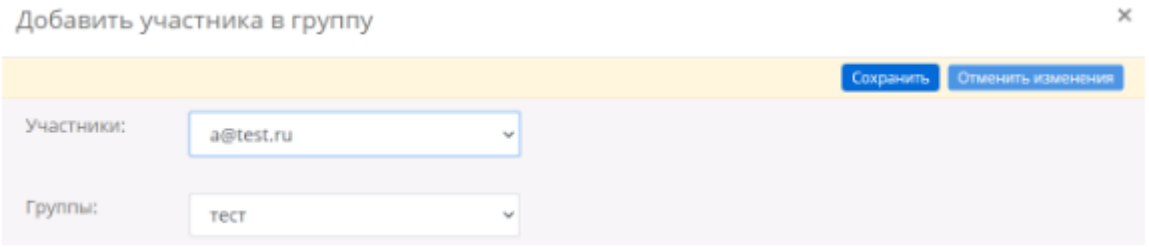
3.2. Создание групп участников

Затем Нужно ввести название группы (лучше придумать “говорящее” название, указывающее на состав участников или соответствующее определённому событию), и затем добавить участников (кнопка “Добавить участника в группу”).

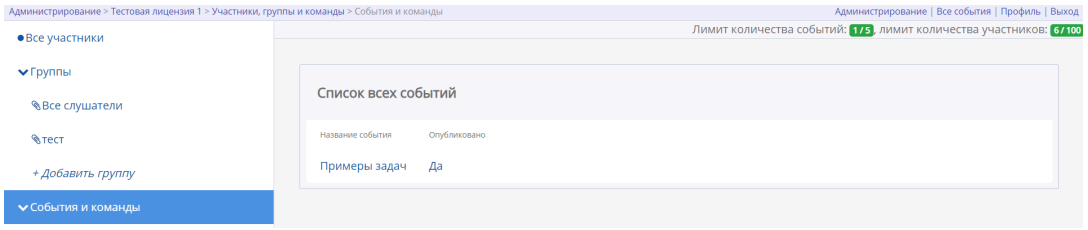


Нужно ввести название группы (лучше придумать “говорящее” название, указывающее на состав участников или соответствующее определённому событию), и затем добавить участников (кнопка “Добавить участника в группу”).





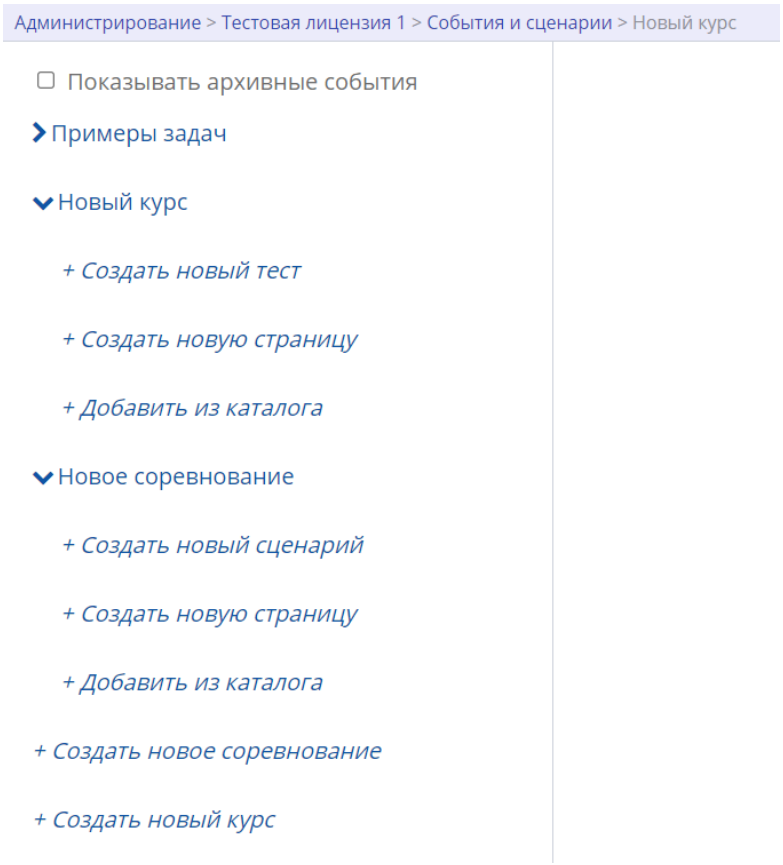
Когда будет создано событие, появится возможность выбрать, какая группа участников сможет в нём участвовать. Список доступных событий можно посмотреть во вкладке “События”.



4. Управление событиями

4.1. Создание событий

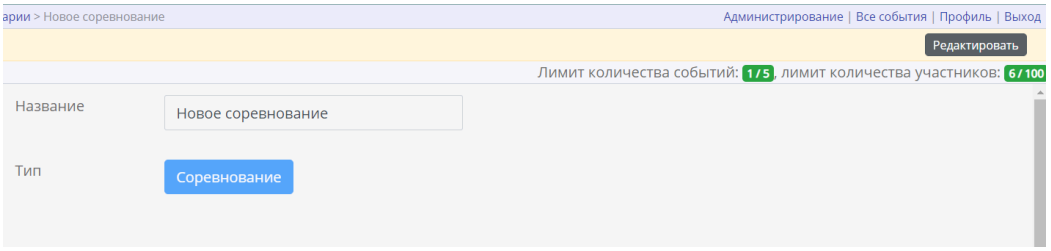
В «Орбита.Челлендж» существует два типа событий: соревнование и курс. Чтобы создавать и редактировать соревнования и курсы, нужно выбрать на панели администрирования “Управление событиями и сценариями”



Соревнование - это событие, которое может содержать в себе сценарии и страницы. К сценариям относятся тест, программа (Python), симуляция, свободное проектирование. Страницы - это блоки, как правило, содержащие теоретическую информацию, которая может быть представлена текстом, изображениями и видеороликами. Задачи в соревнованиях отображаются согласно дате и времени их открытия.

Курс - это событие, элементы которого упорядочены, а задачи открыты всегда. Задачами может выступить либо тест, либо страница. Курс можно использовать, чтобы подвести итог по пройденной теме, резюмировать основные тезисы и формулы, предоставить доступ к дополнительным материалам (например, видеороликам) и проверить теоретические знания участников.

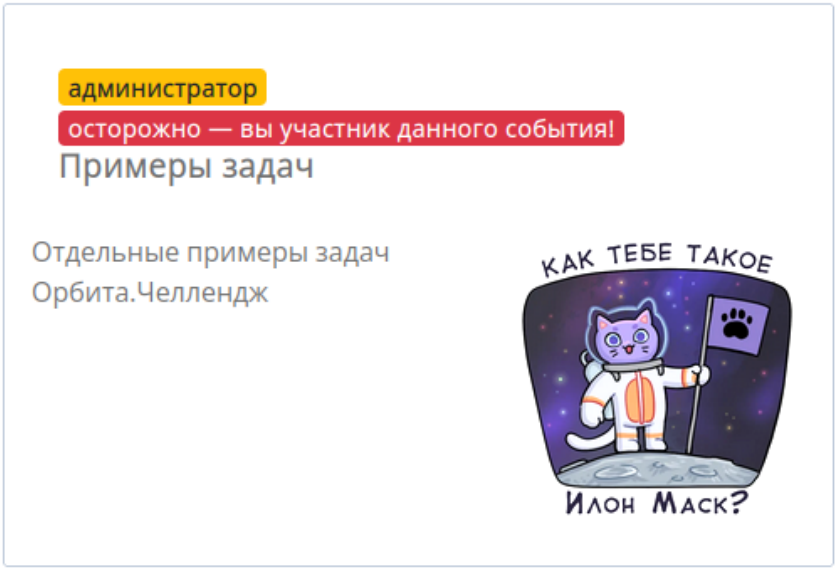
Для создания новых событий служат кнопки “Добавить новое соревнование” и “Добавить новый курс”. Чтобы отредактировать новое событие, необходимо кликнуть на событие и нажать на кнопку “Редактировать” в правом верхнем углу.



Тип события - соревнование или курс - зависит от того, какое событие было создано; изменить его в процессе нельзя. Событие состоит из карточки события и элементов события (страниц и сценариев).

4.1.1. Карточка события

На главной странице сайта события представлены в виде карточек события с описанием и указанием времени завершения. По нажатию на карточку события осуществляется переход на страницу с элементами события.

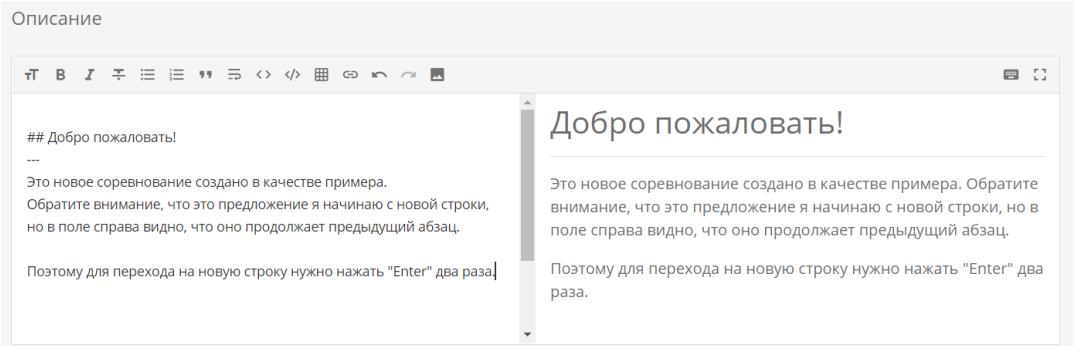


Карточка события содержит в себе:

- название события
- описание
- картинку
- дату начала и окончания
- заголовок и текст приветствия

4.1.1.1 Описание события

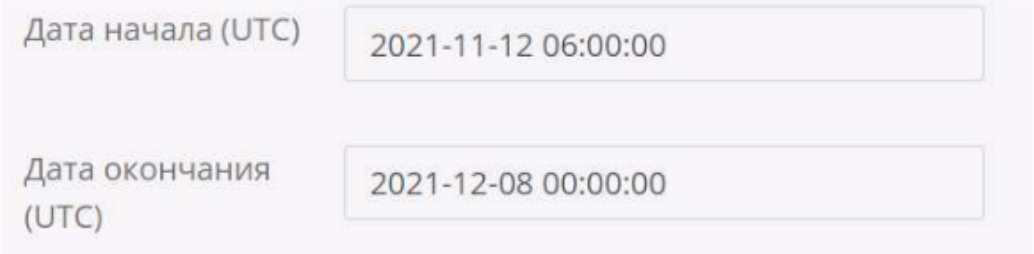
Описание и всплывающий в новом окне текст приветствия, а также все остальные текстовые блоки записываются во встроенном редакторе, где можно настраивать форматирование, добавлять таблицы, ссылки, изображения и видеофайлы. После внесения текста в поле слева, справа отобразится его вид для пользователя. Обратите внимание, что для перехода на новую строку при печати текста необходимо нажать клавишу переноса строки дважды.



Также к описанию можно добавить картинку: из базы картинок, которую можно пополнять, загружая изображения со своего ПК, или по URL картинки, найденной в интернете (чтобы узнать URL любой картинки, можно нажать на неё правой кнопкой мыши и выбрать “Копировать URL картинки”).

4.1.1.2. Дата начала и окончания события

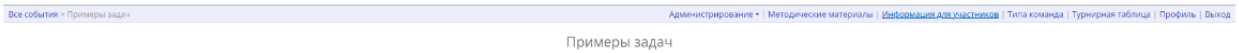
Если создано соревнование, то следует определить дату и время его начала и окончания.



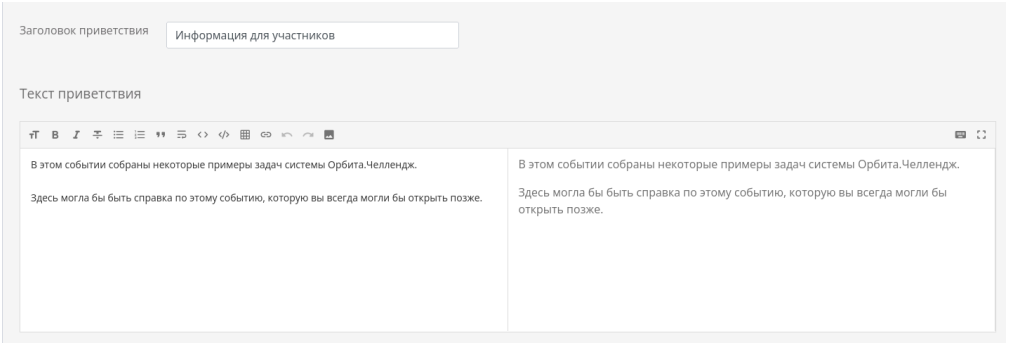
Дата задаётся в формате год-месяц-число, а время указывается по шкале всемирного координированного времени в формате часы:минуты:секунды.

4.1.1.3. Приветствие

Приветствие - сообщение, появляющееся в отдельном всплывающем окне при переходе к элементам события или при нажатии на ссылку “Информация для участников” в верхнем правом углу на странице с элементами события.

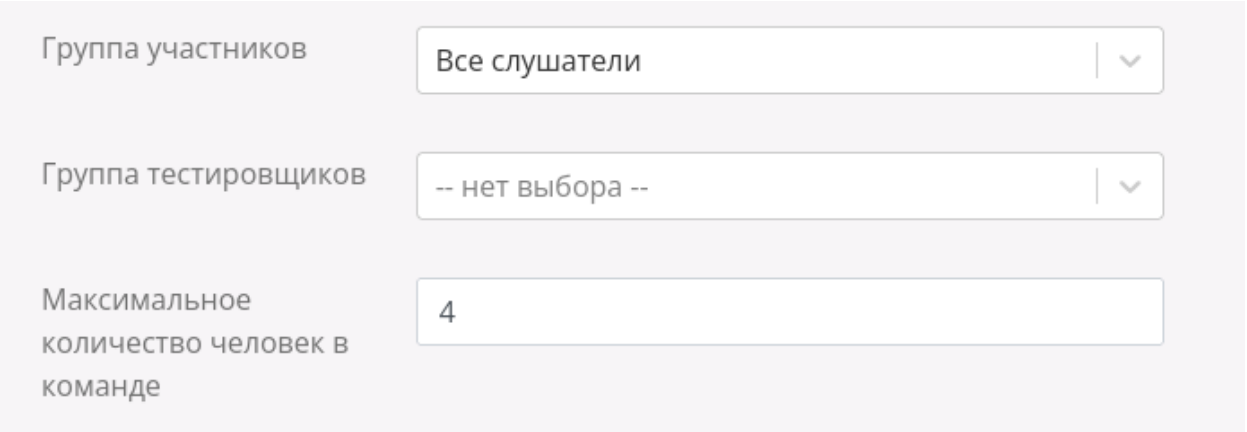


В настройках отдельно задаются заголовок приветствия и его текст. В текст приветствия можно при необходимости добавить изображения, видеофайлы и ссылки с помощью инструментов редактора текста.



4.1.2. Управление доступом участников

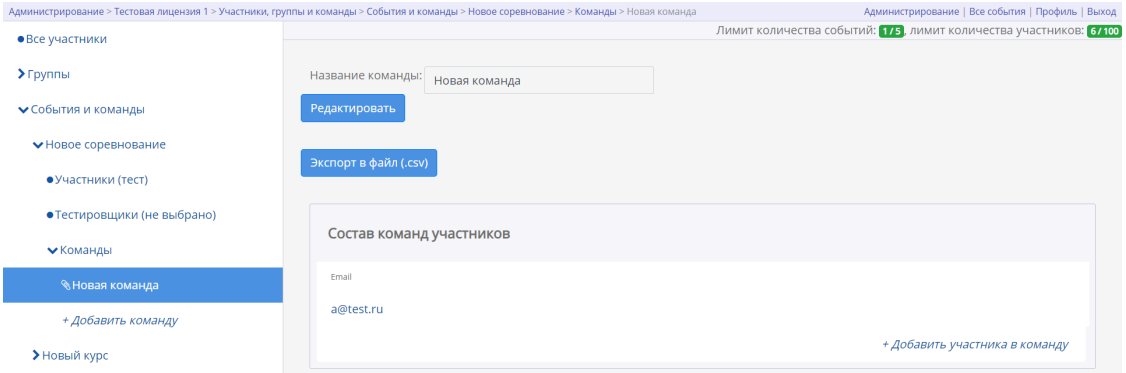
Каждое соревнование или курс можно сделать доступным для ограниченного числа пользователей, выбрав в качестве участников конкретную группу, заблаговременно созданную администратором в разделе “Участники, группы и команды”. Также можно создавать группы тестировщиков, у которых есть неограниченное количество попыток для решения сценария.



4.1.3. Создание команд

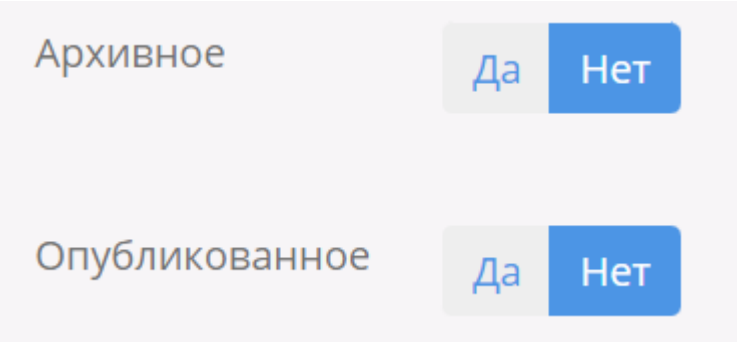
Часто участники работают в команде. В настройках события можно задать максимальное количество человек в команде. После этого в разделе “Участники, группы и команды” можно будет создавать команды и объединять в них участников события. Для этого нужно зайти во вкладку “События и команды”, выбрать интересующее событие, развернуть вкладку “Команды” и нажать “Добавить команду”.

В появившееся поле потребуется ввести название команды, после чего можно нажать “Добавить участника в команду” и во всплывающем окне выбрать нужного участника. Создавать команды, а точнее, добавлять в них участников, можно только после того, как в настройках события выбрана группа участников.



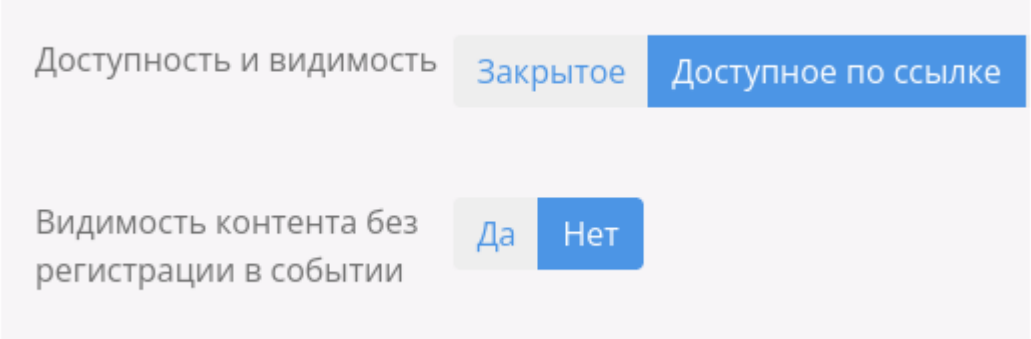
4.1.4. Общий доступ и видимость события

После того, как событие создано, настроено, проверено и туда добавлены необходимые задачи и страницы, можно его опубликовать - изменив свойство “Опубликованное” на “Да”. Также прошедшее или ненужное событие можно не удалять, а сделать архивным.



В настройках события (курса) можно настроить его видимость и доступность.

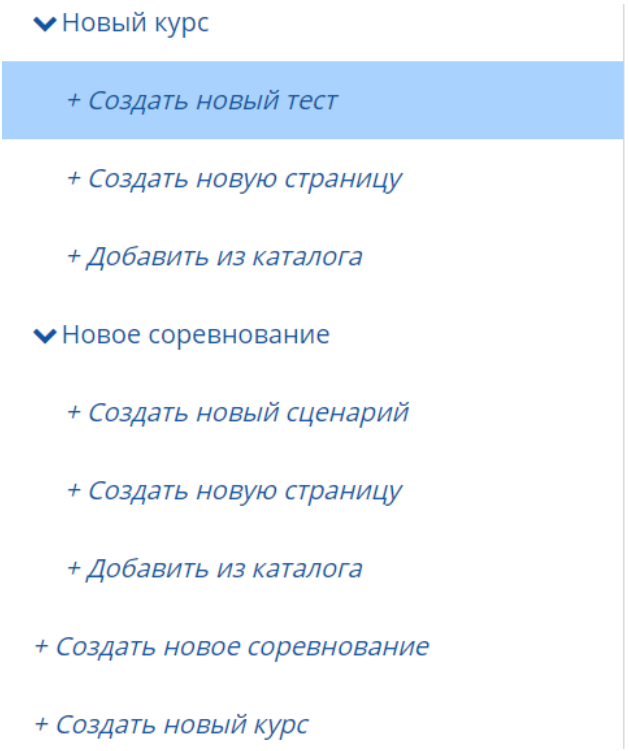
Видимость события	**Описание**
Закрытое	Событие видят и имеют к нему доступ только подключенные администратором участники
Доступное по ссылке	Событие видят только подключенные к нему участники. Если установлен признак “Доступна саморегистрация”, то любой пользователь может зарегистрироваться самостоятельно по указанной секретной ссылке.
Открытое **(устанавливается техподдержкой для согласованных событий межрегионального масштаба)**	Все пользователи видят событие в списке на главной странице. Если установлен признак “Доступна саморегистрация”, то любой пользователь может зарегистрироваться самостоятельно.



4.2. Управление элементами событий

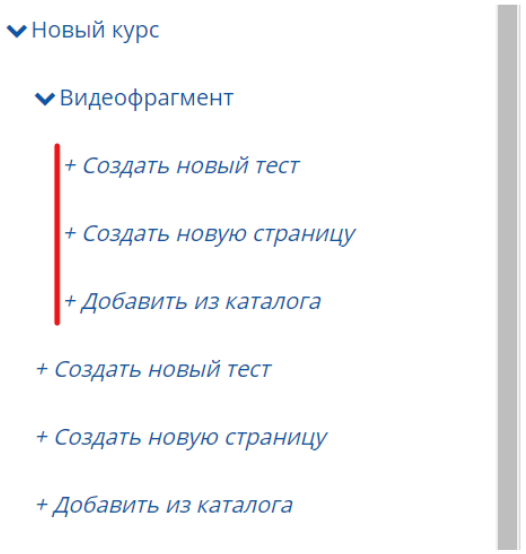
В события можно добавлять элементы: тесты и страницы для курса, сценарии и страницы - для соревнования. Для этого необходимо выбрать

соответствующее действие. Порядок элементов события можно изменять, перетаскивая их мышкой.



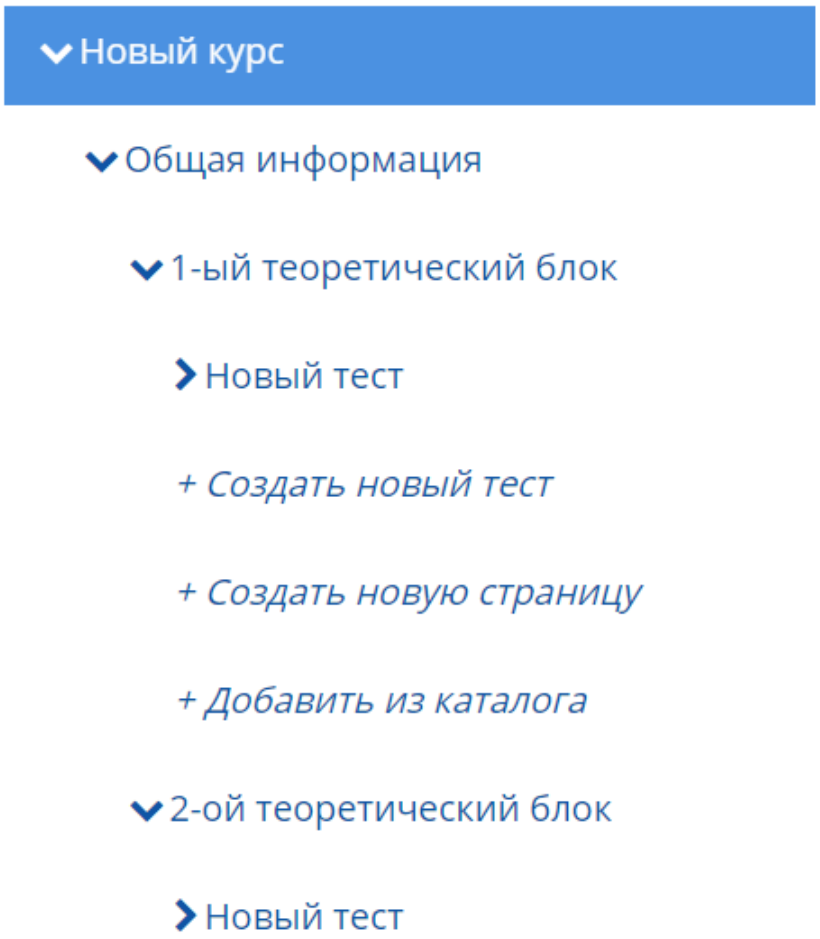
Иерархия элементов событий в курсе

Курс - событие, поддерживающее иерархию элементов. При добавлении новой страницы становится возможным добавление теста или второй страницы, доступ к которым будет осуществляться только после перехода на страницу, стоящую выше в иерархии. Для этого нужно добавлять элемент события, используя ссылки под уже созданной страницей.

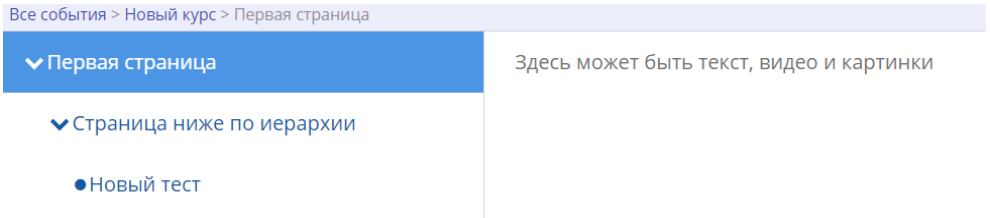


При составлении курса можно руководствоваться следующей логикой: сначала участникам следует узнать или вспомнить теорию, и затем решить

задачу-тест, чтобы закрепить материал. В таком случае тест может располагаться в иерархии под соответствующей ему страницей с теоретическим блоком.

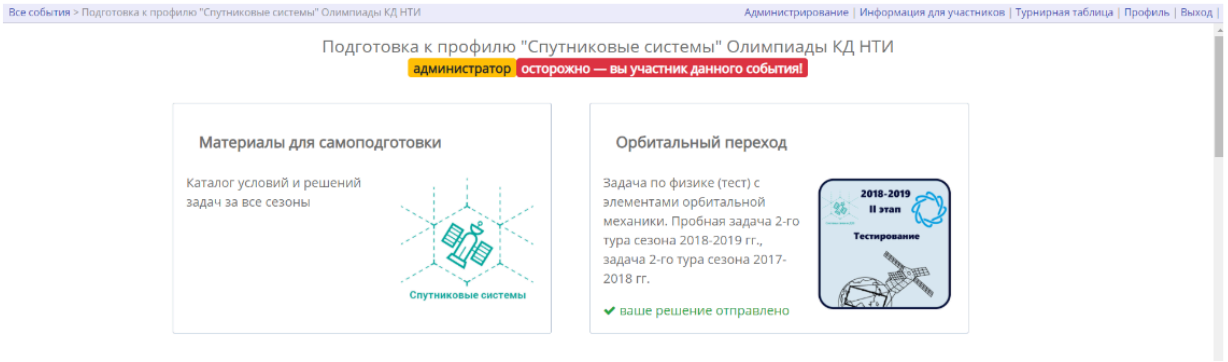


При переходе на страницу курса пользователь сможет перемещаться по иерархии элементов курса, нажимая на ссылки слева.

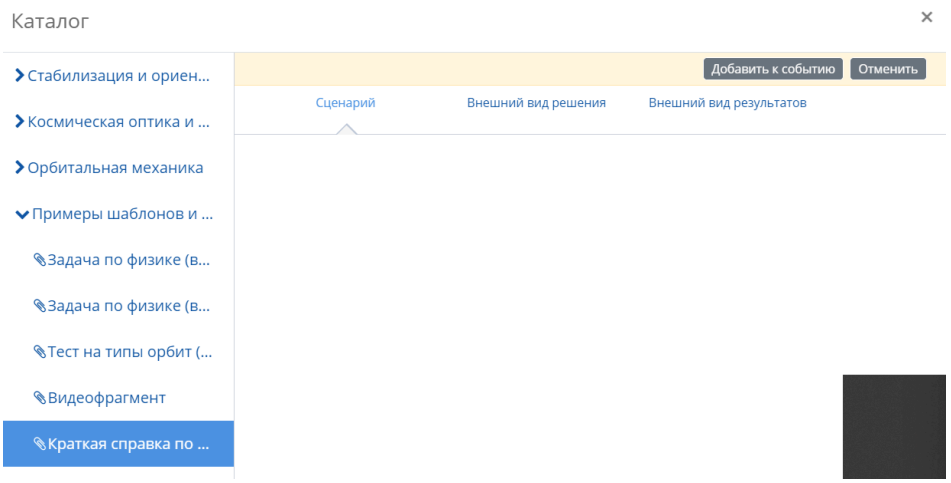


Проведение соревнований

На странице соревнования его элементы выглядят как карточки, расположенные в порядке, определённом администратором. Описания карточек и картинки добавляются при редактировании элементов соревнования.



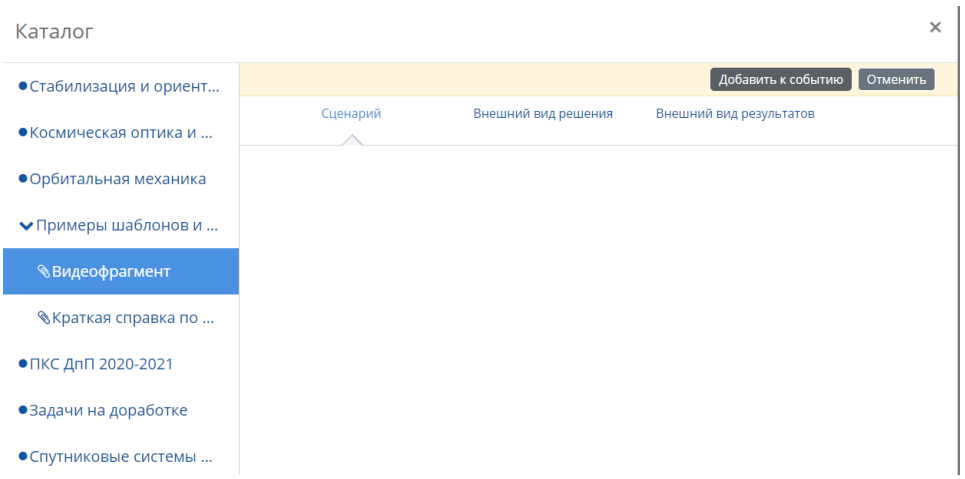
В качестве первого элемента соревнования можно добавить из каталога краткую справку по Орбите, где описано, как работать с симулятором (как посмотреть описание задачи и критерии оценки, отправить решение, посмотреть результаты).



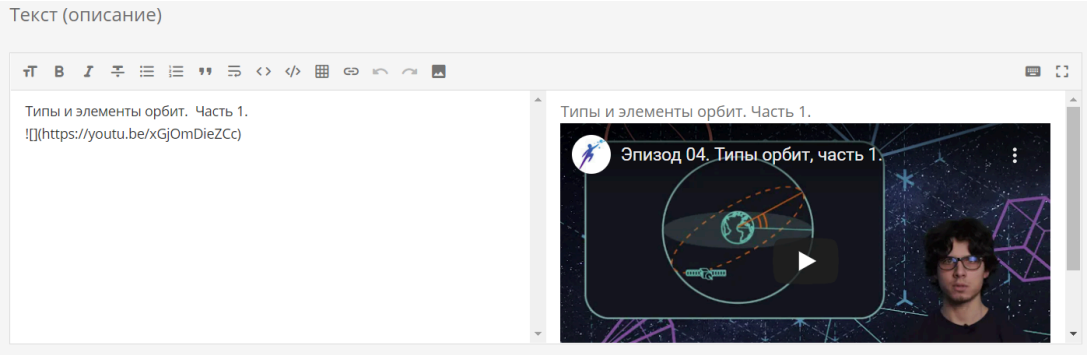
4.2.1. Добавление элементов событий из каталога

В события можно добавлять доступные для используемой лицензии сценарии и страницы из каталога. В каталоге могут содержаться задачи на различные темы, такие как: движение аппарата на орбите, стабилизация и ориентация, проектирование группировки аппаратов, шаблоны тестов (задачи по физике), - а также шаблоны страниц.

Доступные элементы каталога выделены галочкой, недоступные - точкой. Для добавления элемента каталога к событию нужно выбрать интересующий шаблон или задачу и нажать “Добавить к событию”.

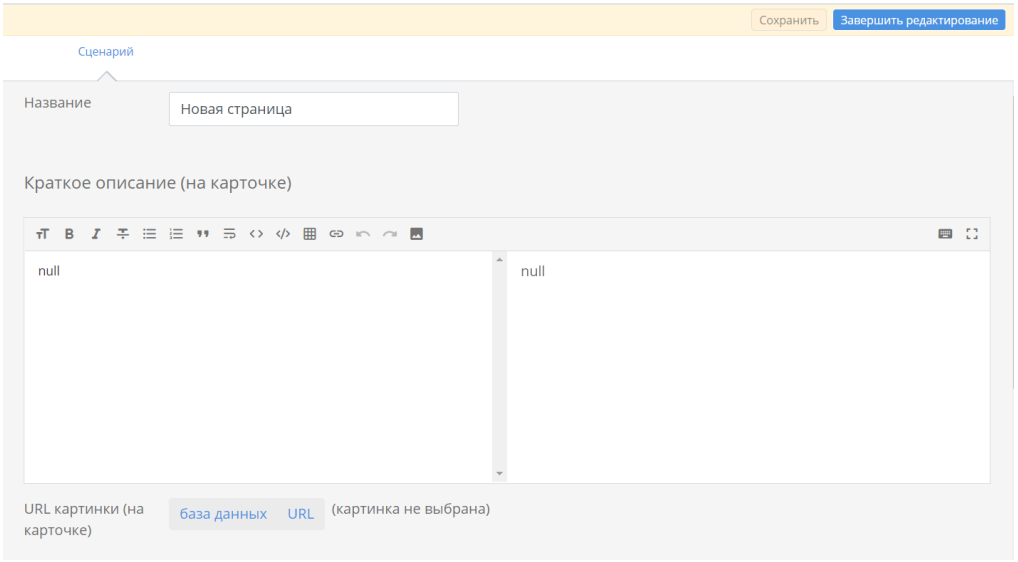


Например, если добавить к событию видеофрагмент и открыть режим редактирования, можно увидеть синтаксис добавления ссылок - сначала ставится восклицательный знак, квадратные скобки, а ссылка записывается в круглых скобках. Стандартную ссылку из шаблона теперь можно поменять на URL видеоролика или картинки. Можно разместить несколько видео и картинок на одной странице.

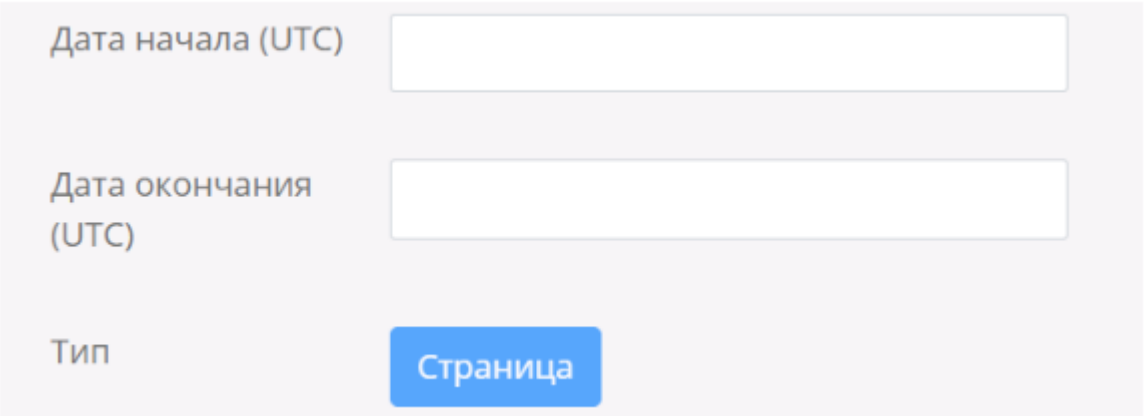


4.2.2. Редактирование элементов события типа “Страница”

При редактировании страниц, добавленных к соревнованию, можно добавить краткое описание и картинку, которые будут располагаться на карточке этой страницы.



Можно определить временные рамки, в которых страница будет доступна для просмотра.



В блок “Текст” (описание) можно добавить любую информацию, видеоролики и картинки. О форматирование текста и добавлении ссылок было сказано раньше в этой главе.

4.2.3. Настройка элементов типа “Сценарий”

Так же, как и для страницы, в режиме редактирования для сценария можно добавить краткое описание и картинку на карточке, дату и время начала и окончания сценария.

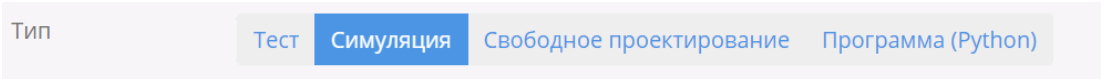
Основные блоки сценария

Выбирая сценарий, пользователь может видеть четыре значка: описание, начальные данные, критерии оценки и решения.

Описание задаётся в блоке “Подробное описание” при редактировании сценария; в него помещается подробное условие задачи.

Начальные данные соответствуют блоку “Описание входных параметров”; в него можно записать константы и значения величин, необходимые для решения данной задачи, а также можно добавить ссылки на API аппарата и вспомогательные материалы.

Критерии оценки - это блок “Описание начисления очков”; в нём следует подробно описать правила начисления баллов: максимальное количество, штрафы, количество попыток. То, какой вид примет форма отправки **решений** и как именно они будут проверяться, зависит от выбранного типа сценария.



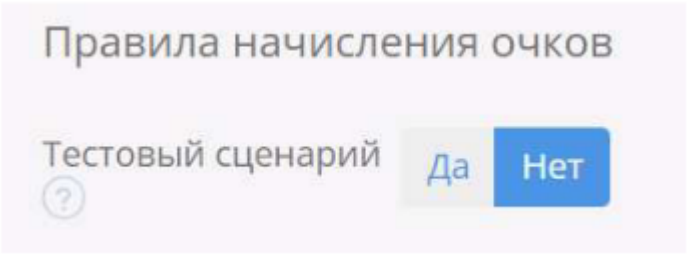
Типы сценариев

Представлено четыре типа сценариев: тест, симуляция, свободное проектирование и программа (Python).

1) **Тест** - задача, в решении которой требуется вписать ответ в виде числа или выбрать один или несколько предложенных вариантов ответа.
2) **Симуляция и свободное проектирование** предполагают написание участником программы на языке JavaScript.
3) **Программа (Python**) - задача на написание программы на языке “Python”, отвечающей требованиям, поставленных в описании.

Начисление очков

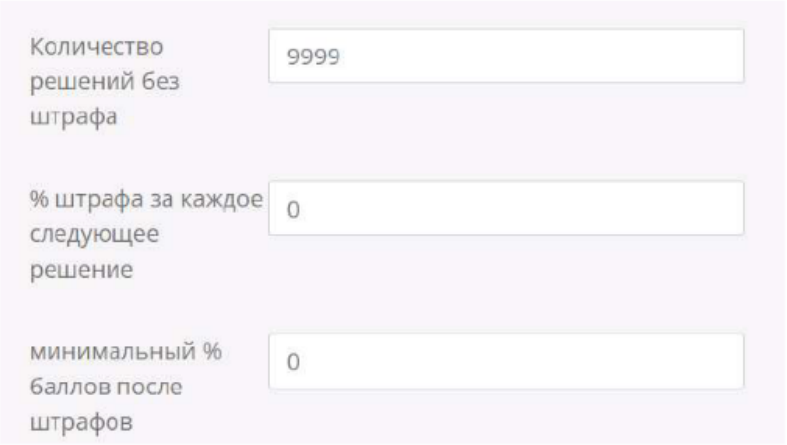
В блоке “Правила начисления очков” нужно задать, является ли сценарий тестовым. Если да, то участникам вместо баллов будут начисляться “Очки успеха” - по ним можно понять, как участник справился с поставленной задачей, но на итоговый результат при соревновании между командами они не влияют. Можно создать один или несколько тестовых сценариев, с которых участники начнут освоение симулятора “Орбита”, а соревновательная часть будет проводиться с помощью других, более сложных, сценариев, когда все участники в достаточной степени изучат работу симулятора.



При вычислении заработанных баллов можно установить штраф - процент, на который будет уменьшаться максимально возможное количество баллов в зависимости от номера попытки. Особенно полезно устанавливать штрафы, чтобы участники не могли решить тест перебором вариантов.

Чтобы настроить штраф требуется:

- задать количество решений, которые участники могут отправить без штрафа;
- задать процент штрафа за решения сверх этого количества, отнимаемый от полученного балла;
- задать минимальный процент баллов, который может остаться после вычитания штрафов независимо от количества попыток.



Далее нужно определить, требует ли задача ручной оценки администратора. Решения всегда можно посмотреть в разделе “Просмотр решений” на панели администрирования, а скорректировать баллы - в разделе “Корректировка оценок - штрафы и бонусы командам и участникам”.

Также нужно определить, является ли сценарий индивидуальным. Если участник соревнования состоит в какой-либо команде, то при решении индивидуального сценария баллы за него получит только конкретный участник, а не вся команда.

Дополнительно можно установить максимальную разницу попыток внутри команды. Если сценарий не является индивидуальным, то баллы за решение будут

начисляться всей команде, но можно ограничить попытки каждого члена команды таким образом, чтобы решения не отправлял один и тот же участник команды.

Также можно определить, возможность отправки только одного решение (например, при решении теста с выбором предложенного ответа участник рано или поздно укажет верный ответ).

Редактор форм ввода

В большинстве типов сценариев для создания формы ввода решения пользователем может быть использован редактор форм ввода.

Форма ввода может редактироваться в визуальном или ручном режиме, через файл описания в формате JSON.

При редактировании в визуальном режиме файл описания создается или изменяется автоматически.

Поля в файле перечисляются по очереди в формате

```
[
 {
   Свойство поля1: "Значение",
   Свойство поля1: "Значение",
   Свойство поля1: "Значение"
 },
 {
   Свойство поля2: "Значение",
   Свойство поля2: "Значение",
   Свойство поля2: "Значение"
 },
]
```

Например

```
[
 {
  "field": "staticDescription",
  "defaultValue": "Введите массу топлива в киллограммах",
  "type": "html"
 },
 {
  "exclusiveMin": 0,
  "unit": "кг",
  "required": true,
  "field": "mass",
  "type": "double",
  "name": "Масса"
 }
]
```

Поддерживаемые типы полей:

Тип в JSON	Описание типа	Примечания
check	Да / Нет (флажок)	
date	Дата	
double	Произвольное число	
enum	Выбор варианта из перечня	
file	Загрузка файла	
html	Примечание в фиде форматированного как HTML текста	Ввод данных недоступен
image	Выбор или указание изображения	
int	Целое число	
javaScriptEditor	Программа на JavaScript	
pythonEditor	Программа на Python	
string	Текст (одна строка)	
text	Текст (многострочный)	

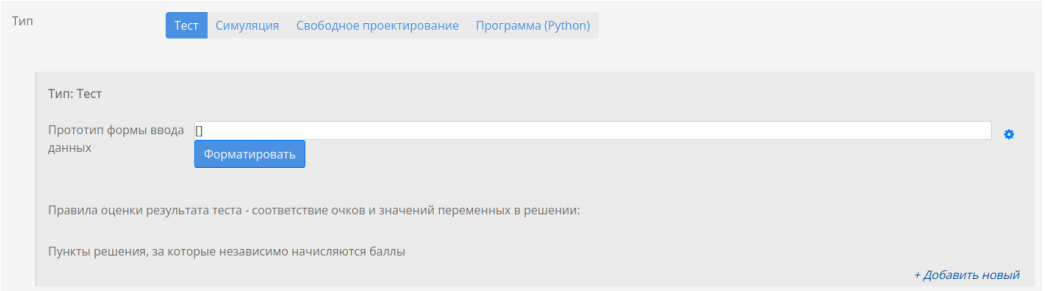
Параметры полей

Параметр	Описание	Возможные значения	К каким полям применимо
name	Заголовок поля, который видит пользователь	Строка текста в кавычках	Все поля
type	Тип поля (из списка выше)	Строго из списка выше, в кавычках	Все поля
field	Переменная, куда сохраняется введенное значение (если ввод разрешен)	Имя переменной (латиница, цифры и подчеркивание без пробелов), в кавычках	Все поля
disabled	Поле недоступно для редактирование	true или false	Все поля
info	Всплывающая при наведении подсказка	строка в кавычках	Все поля
required	Необходимое для заполнения поле	true или false	Все поля ввода, кроме флажков
mandatory	Необходима установка этого флажка (напрмер подтверждение согласия с правилами)	true или false	Только для флажков
min	Нижняя граница допустимого диапазона значений (значение может быть равно этой границе)	число	Числовые поля (int, double)
max	Верхняя граница допустимого диапазона значений (значение может быть равно этой границе)	число	Числовые поля (int, double)

exclusiveMin	Нижняя граница допустимого диапазона значений (значение НЕ может быть равно этой границе)	число	Числовые поля (int, double)
exclusiveMax	Верхняя граница допустимого диапазона значений (значение НЕ может быть равно этой границе)	число	Числовые поля (int, double)
unit	Единица измерения	строка в кавычках	Числовые поля (int, double)
defaultValue	Значение по умолчанию	число или строка в кавычках	числовое или текстовое поле (int, double, string, text)
stuff	Элементы списка	массив	только поле выбора из списка (enum)
maxFileContentsKB	Максимальный допустимый размер файла в килобайтах	число	только поле загрузки файла (file)
allowUpload	Разрешить загрузку	true или false	только поле загрузки файла (file)

4.2.4. Сценарий типа Текст (Простой)

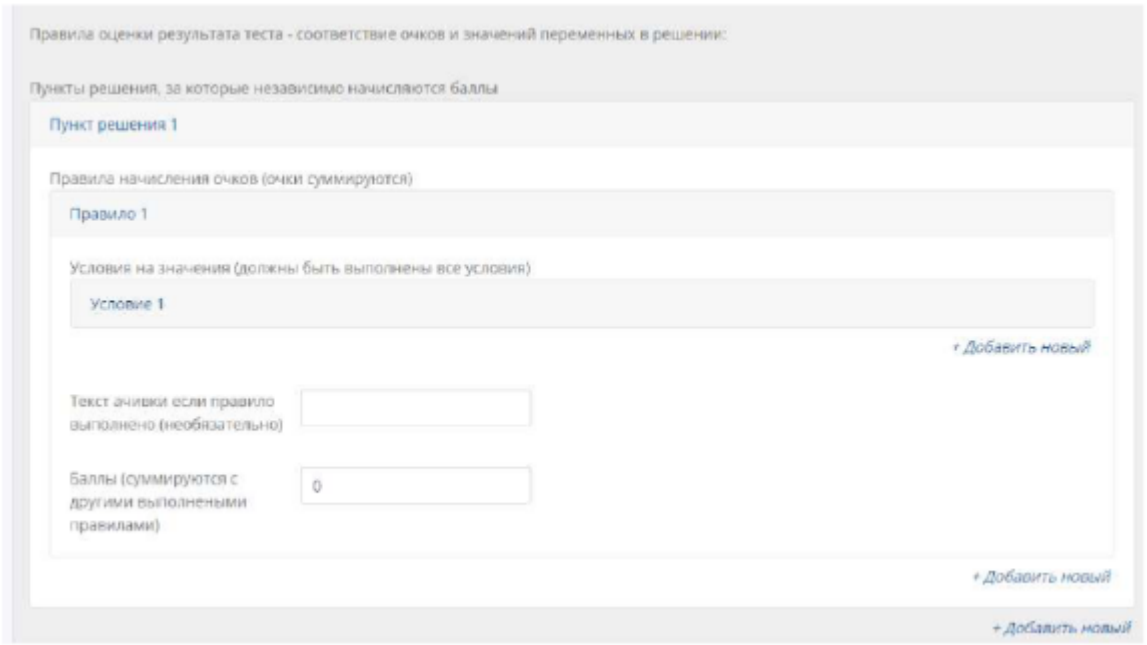
После выбора типа “Тест” для сценария появится поле редактирования теста, содержащее в себе прототип формы ввода данных и правила оценки результата.



Настройка формы ввода данных

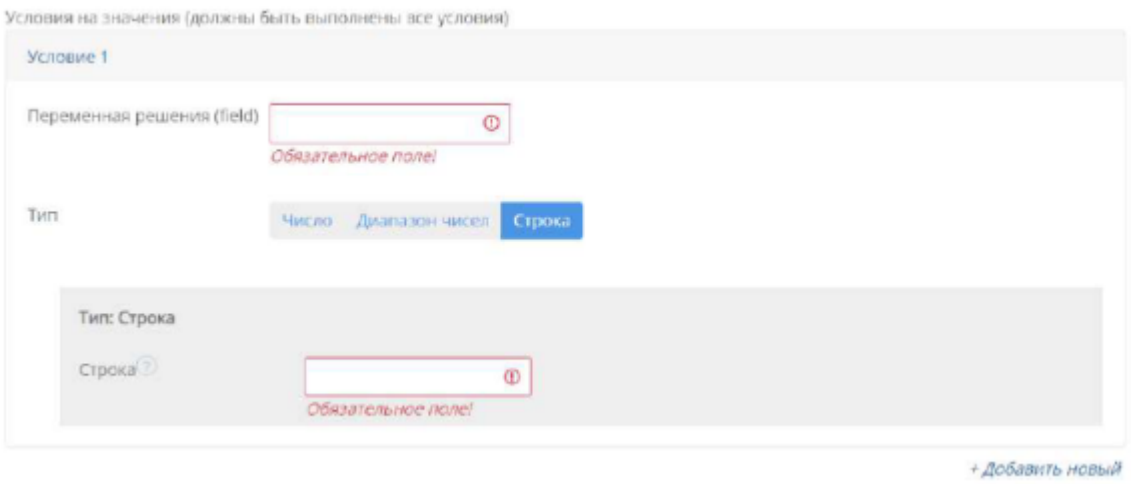
Основная статья: Редактор форм ввода

Для сценариев типа “Тест” можно явно указывать критерии автоматического начисления баллов по одному или нескольким пунктам задачи, в том числе по наборам одновременно выполняющихся условий. Для этого с помощью нажатия кнопки “Добавить новый” в разделе “Пункты решения, за которые независимо зачисляются баллы” необходимо создать “Пункт решения”, в котором должно быть добавлено хотя бы одно “Правило”, в котором будет задано хотя бы одно “Условие”. Каждое выполненное условие добавляет указанный администратором балл и/или комментарий к решению.



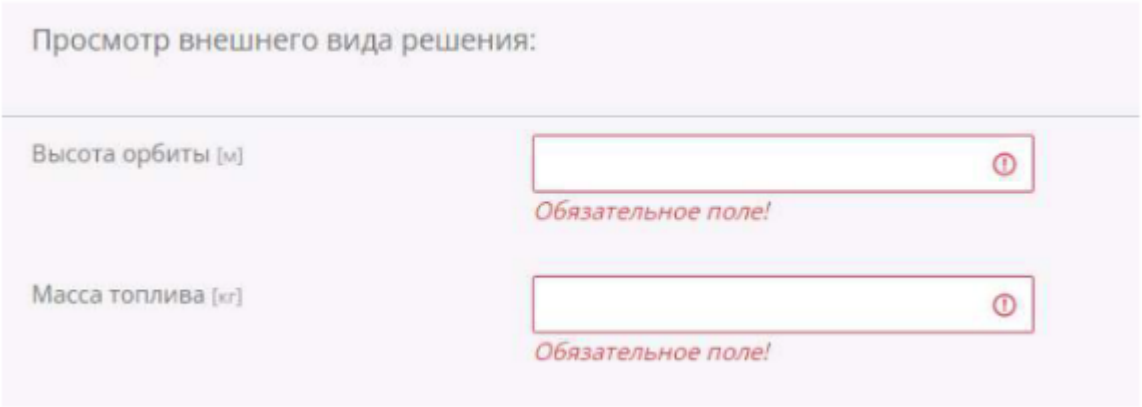
Условия бывают трёх типов:

- значение переменной решения (field) строго равно заданному числу
- значение переменной решения находится в определённом диапазоне чисел
- значение переменной решения совпадает со строкой (текстовым полем), заданной с точностью до регистра и незначащих символов до и после текста (пробелов, знаков препинания, переводов строки)

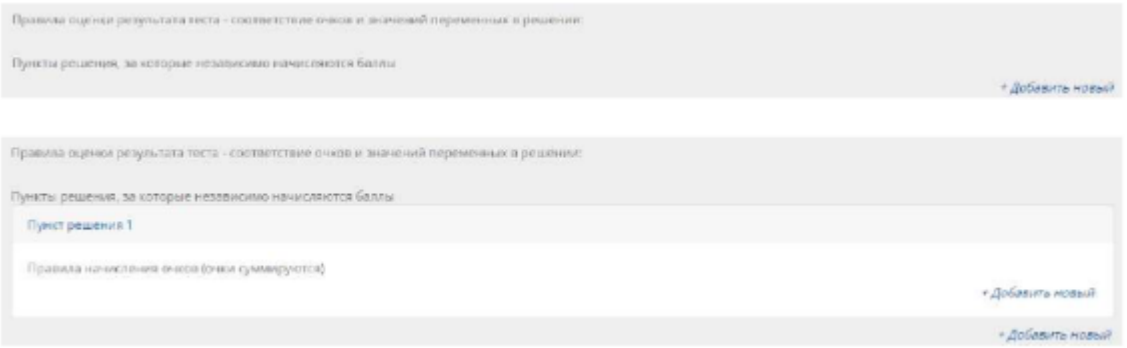


Проверка решения при вводе числового ответа

Для примера возьмём случай, когда участнику соревнования требуется рассчитать сразу несколько величин: например, высоту орбиты и массу топлива.

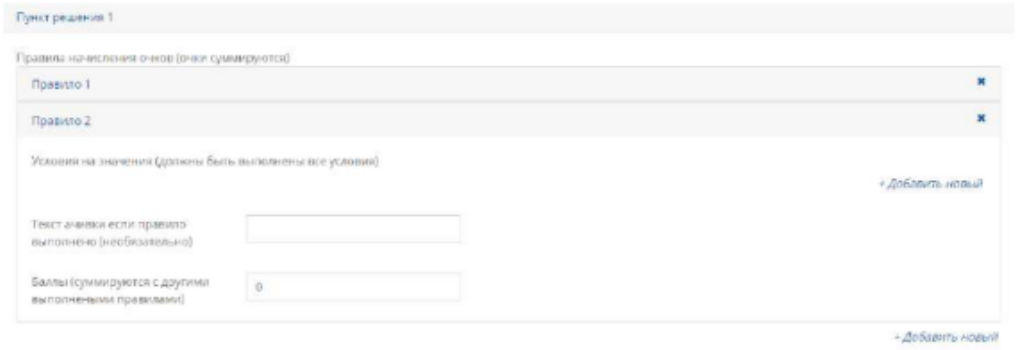


Сначала необходимо добавить новый пункт решения.

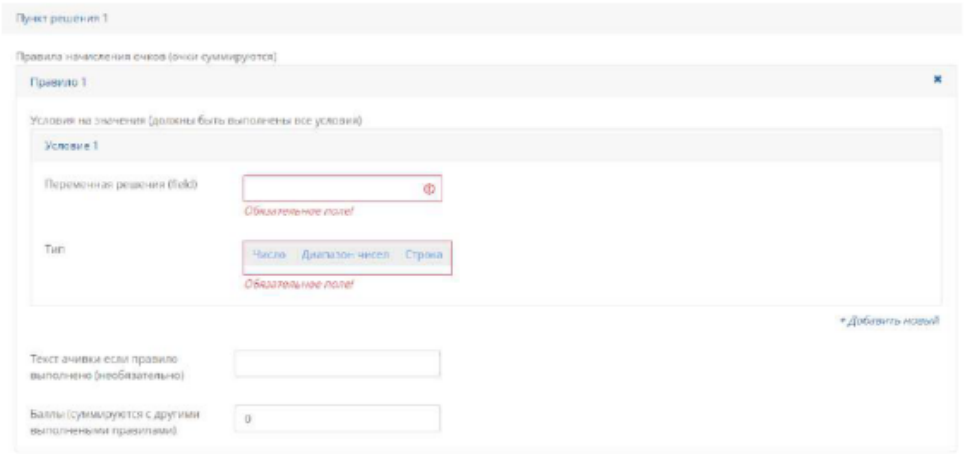


В созданный Пункт решения 1 необходимо добавить хотя бы одно правило. При выполнении сразу нескольких правил заработанные участником очки будут суммироваться; поэтому, если предполагается, что максимальный балл за решение состоит из балла за верное определение высоты орбиты и балла за

верное определение массы топлива, для нашего примера потребуется создать два правила. Если же вы хотите считать верным ответом только тот случай, когда обе величины определены верно, необходимо создать одно правило, а в нём - два условия.



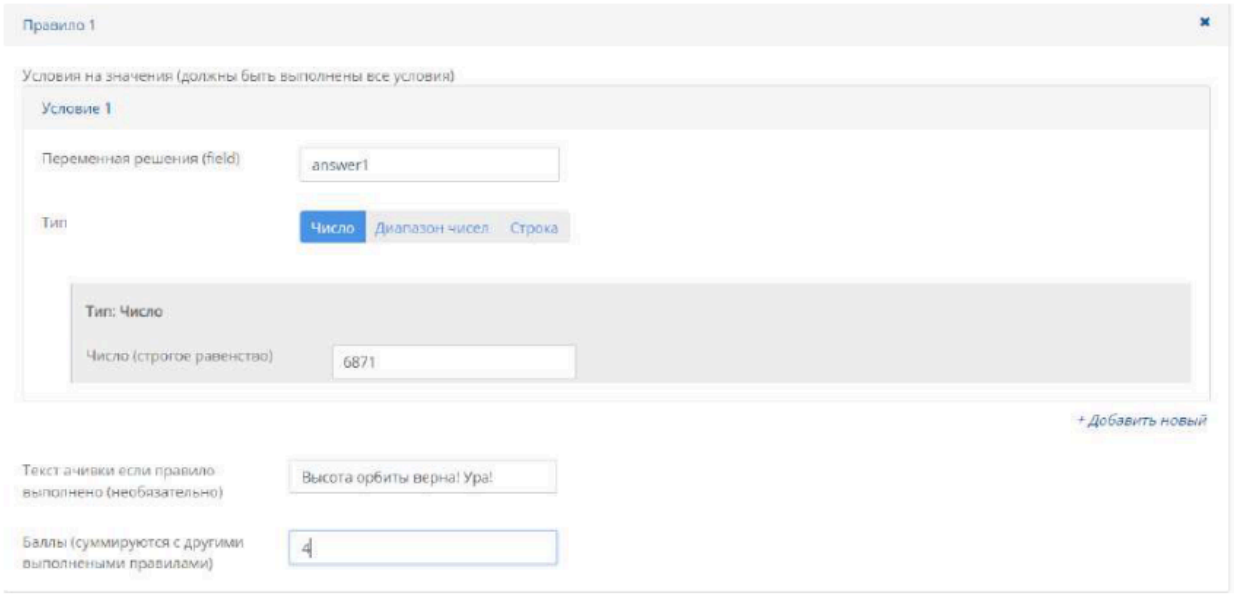
После добавления правила внутри него можно добавить новое условие.



Поле “Переменная решения” (field) - это значение свойства объекта “field”, которое было задано при создании прототипа формы ввода данных. Они должны быть разными у разных объектов; пусть у высоты орбиты используемое в программе имя будет “answer1”, а у массы топлива - “answer2”.

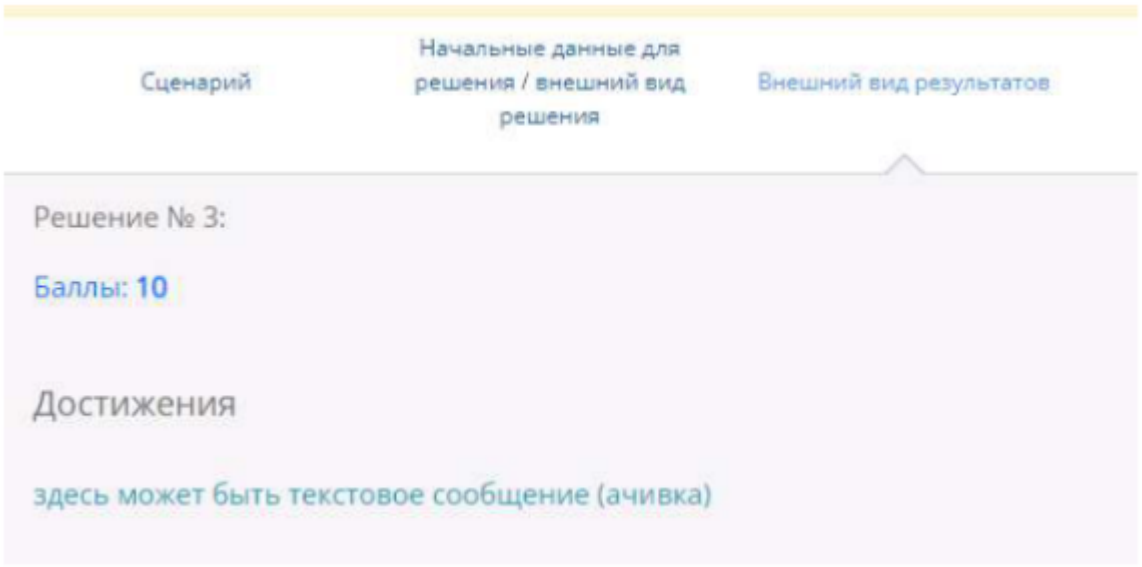
```
{
  "Canvas 0": [
    {
      "field": "answer1",
      "name": "Высота орбиты",
      "type": "int",
      "unit": "м",
      "required": true
    },
    {
      "field": "answer2",
      "name": "Масса топлива",
      "type": "int",
      "unit": "кг",
      "required": true
    }
  ]
}
```

В условии для первого правила запишем переменную решения, соответствующую высоте орбиты. Если высоту орбиты (или любую другую величину, которую требуется посчитать по условию задачи) вычислить достаточно легко, то в качестве типа условия можно выбрать “Число”. Появится поле, куда нужно вписать правильный ответ - проверка решения будет происходить на строгое равенство данного участником ответа правильному.

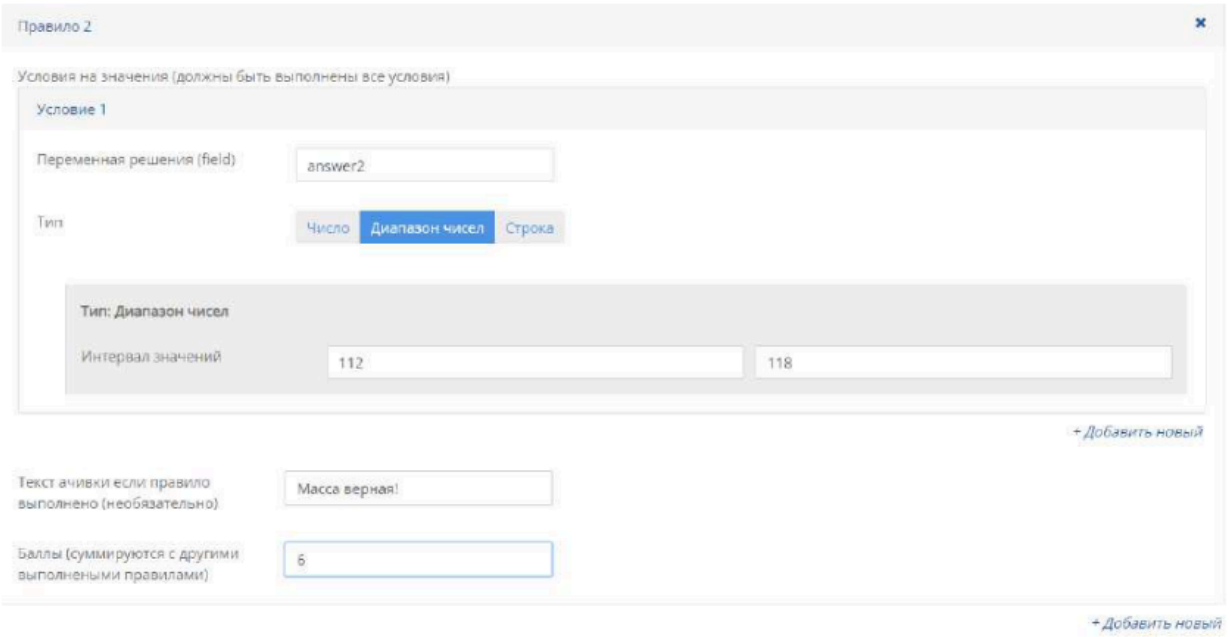


Теперь в правиле начисления очков можно добавить количество баллов, которое будет начислено за верное вычисление высоты орбиты, и при желании написать текст ачивки - сообщения о заработанном достижении, которое участник увидит

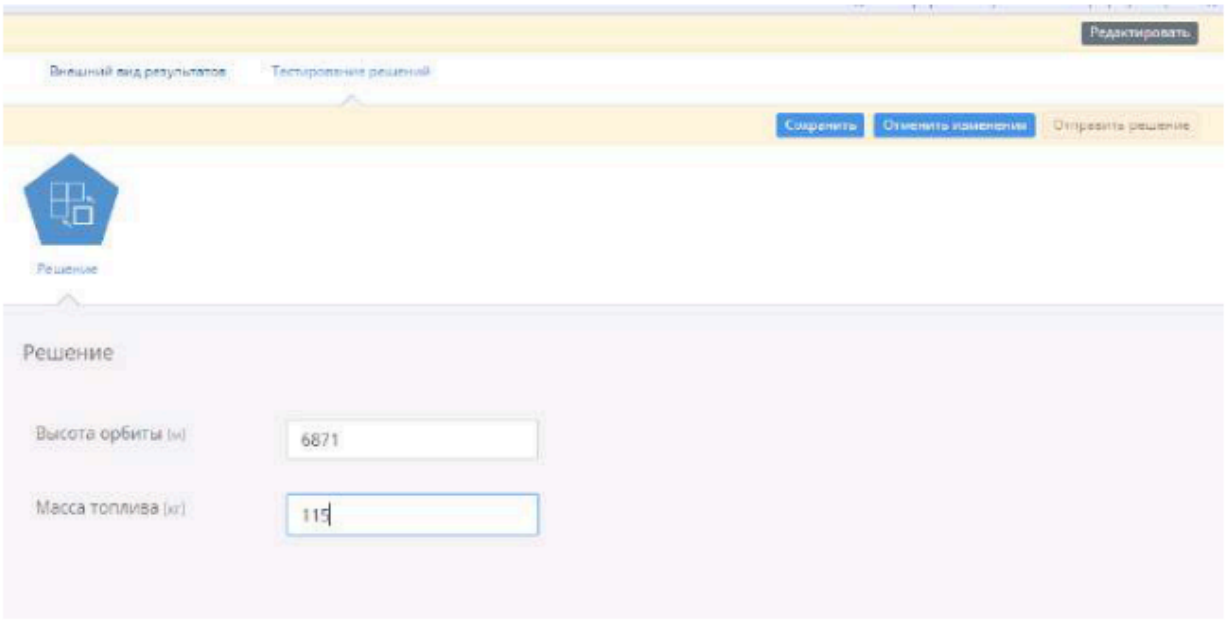
после отправки решения - оно отображается ниже количества заработанных баллов.

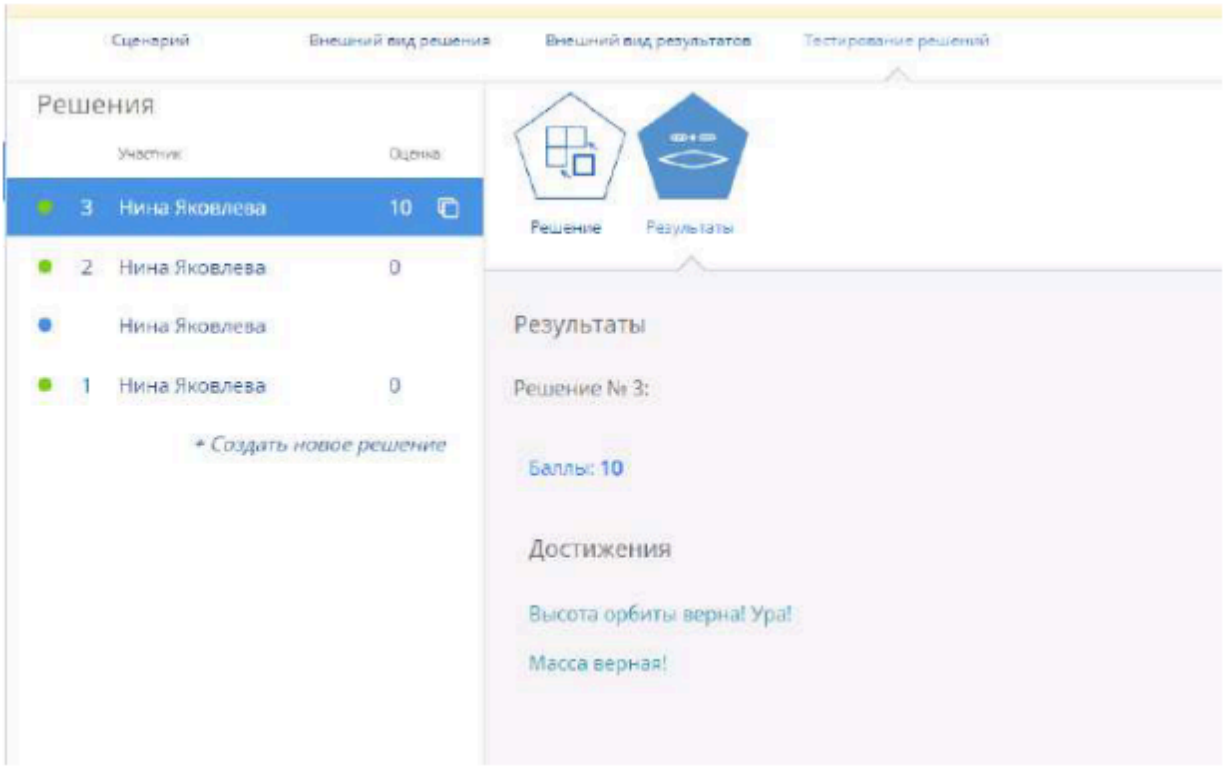


Во втором правиле также нужно настроить количество баллов за правильный ответ, задать текст ачивки и добавить новое условие. Если рассчитать значение величины довольно сложно или ответ может содержать в себе погрешность, то в качестве типа условия можно выбрать диапазон чисел. В приведённом примере верное значение массы топлива - 115 кг с погрешностью ±3 кг.



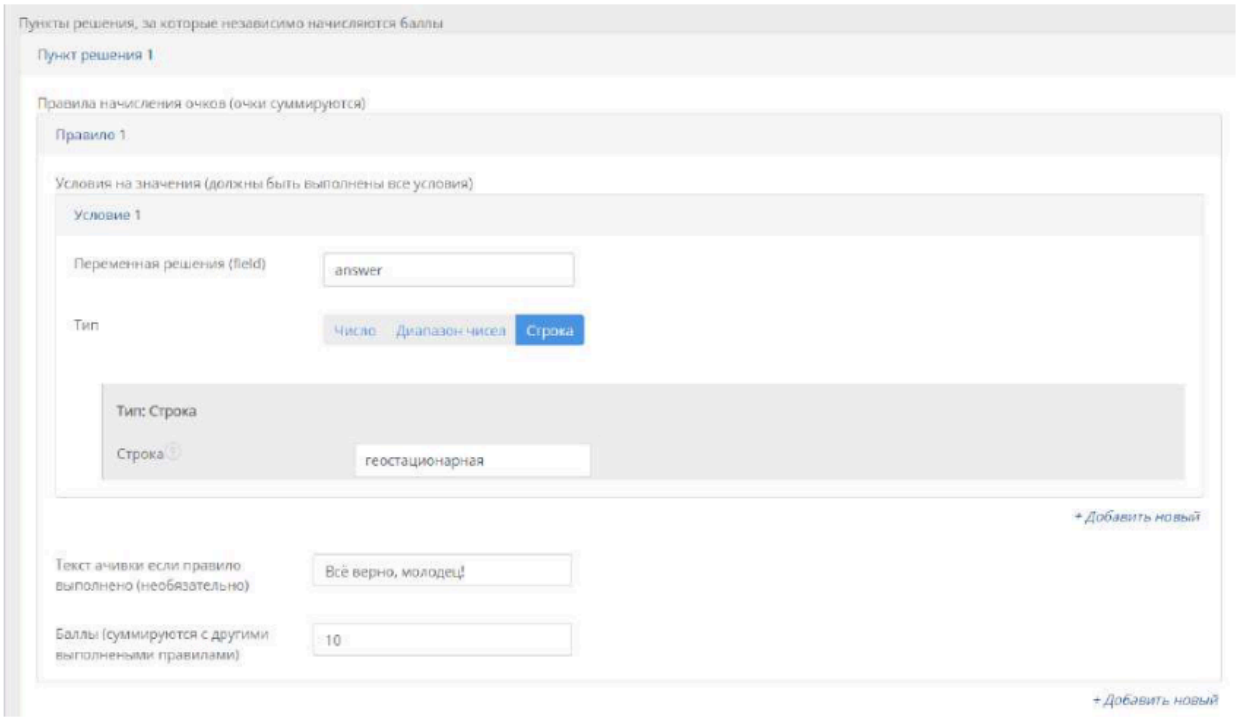
Максимальное количество баллов, которое получит участник, верно рассчитавший и высоту орбиты, и массы топлива - 10 баллов (4 балла + 6 баллов), а также он увидит две ачивки. Можно сохранить изменения редактирования, и тогда можно будет протестировать решение созданного теста во вкладке “Тестирование решений”.





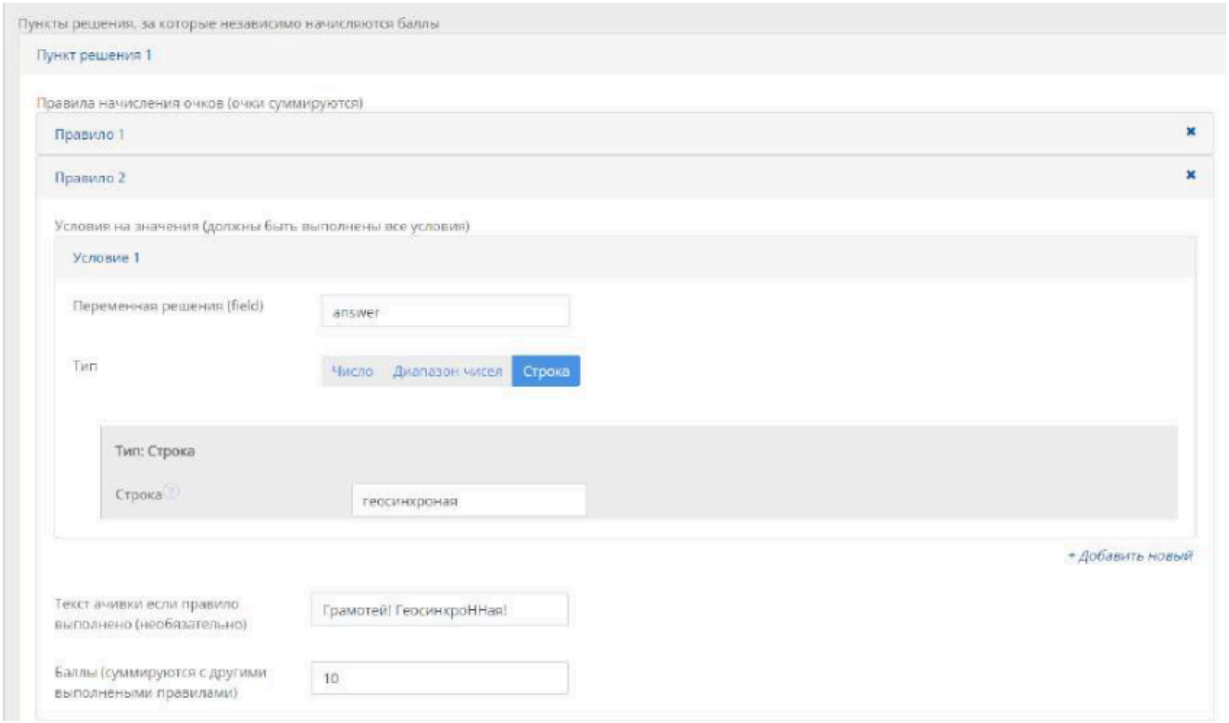
Проверка решения при вводе текстового ответа

Добавьте пункт решения, правило и условие. Запишите в поле “Переменная решения” используемое в программе имя (field) объекта типа “string”, созданного вами в прототипе формы ввода данных. Выберите тип условия “Строка” и введите верный ответ в появившееся поле.



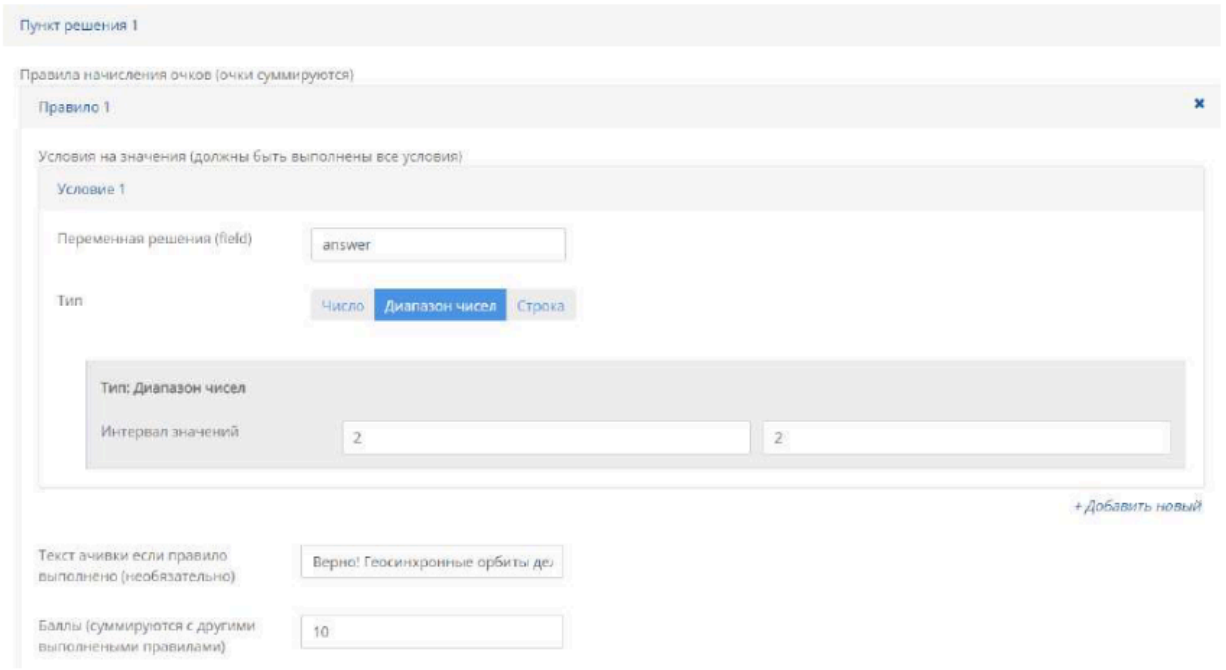
Текстовые ответы могут вызвать у участников соревнования проблемы, если они совершат грамматические ошибки, случайно поставят пробел (который нельзя будет увидеть при визуальной проверке) или напишут слово с заглавной буквы, когда администратором выбран верный ответ из строчных букв. Чтобы избежать возможных ошибок, можно добавить к созданному пункту решения дополнительные правила, где в условии в качестве верного ответа будут даны другие формы написания слова-ответа.

Не путайте создание нового правила с созданием нового условия - несколько условий должны выполняться одновременно, чтобы ответ был засчитан как верный.



Проверка решения при выборе ответа из нескольких вариантов

Если в прототипе формы ввода данных было создано перечисление (enum), то в условии для проверки решения необходимо указать используемое в программе имя перечисления (field) и сравнивать его с одним из существующих значений (value) созданного в нём массива. Если обратиться к примеру в подразделе “Выбор из нескольких вариантов”, то верному варианту “Геосинхронная” орбита соответствует значение, равное 2. В таком случае условие правила начисления очков может выглядеть следующим образом:



Также можно выбрать в качестве типа условия “Число”.

4.2.5. Сценарий типа Тест (Python)

В сценариях типа Тест (Python) задания могут позволять вводить участниками решения, состящие из одного или нескольких значений типа число, строка, многострочный текст, выбор варианта, флажок, файл (в том числе различных типов).

Для создания формы ввода в задаче можно воспользоваться Редактором форм ввода

Основная статья: Редактор форм ввода

Особенностями данного типа заданий являются

1. Оценка результата программой на Python (визуально настраиваемые критерии оценки недоступны, но с помощью программы можно оценивать с любой гибкостью)
2. Условие может содержать случайно определенные для участника параметры.

Для использования в условии случайных параметров необходимо переменную из программы оценки вставить в текст условия в фигурных скобках, например

У Васи {X} яблок и у Пети {Y} яблок, сколько у них яблок всего?

Переменная или переменные, в которых участник вводит ответ, определяются шаблоном ввода данных - параметр field, например

```
[
 {
  "field": "apples_num",
  "name": "Количество яблок всего",
  "type": "int",
  "required": true,
  "unit": "яблок",
  "min": 0,
  "max": 18000000
 }
```

В визуальном редакторе у соответствующего поля эта параметр "Переменная", в данном случае это *apples_num*

Программа оценки должна содержать функции

- generate() - здесь задаются параметры, если это необходимо
- check(parameters,inputs) - здесь оцениваются параметры

Функция check должна возвращать структуру, в которой могут быть следующие поля:

Поле	Назначение	Примечание
score	Начисленные очки	Целое число, обязательное
message	Присвоенное "достижение"	Строка, не обязательное
log	Записи лога	Многострочный текст, не обязательное

Пример программы оценки

Пример программы оценки, которая X и Y устанавливает в случайные целые значения от 1 до 20.

```
import random


def generate():
    # Сохраняем данные о генерации в логе для разработчика задачи
    print("generate")
    # Возвращаем параметры X и Y из условия, при этом устанавлваем их случайными от 1 до 10
    return {
        "X": random.randint(1,20),
        "Y": random.randint(1,20)
    }



def check(parameters, inputs):
    # Сохраняем данные о проверке в логе для разработчика задачи
    print("check")
    # Ответ верен, если введенное значение apples_num равно сумме X и Y
    is_correct = parameters["X"] + parameters["Y"] == inputs["apples_num"]
    # Если ответ верен, вернем 10 очков и, при желании, добавим комментарий
```

```
if is_correct: return {"score": 10, "message": "Верно, всего у Пети и Васи именно столько яблок"}

    # Иначе вернем 0 очков и, при желании, добавим комментарий (можно добавить elif для дополнительных градаций)

else: return {"score": 0, "message": "Не совсем"}
```

4.2.6. Сценарий типа Тест “Симуляция”

Создание прототипа формы ввода данных

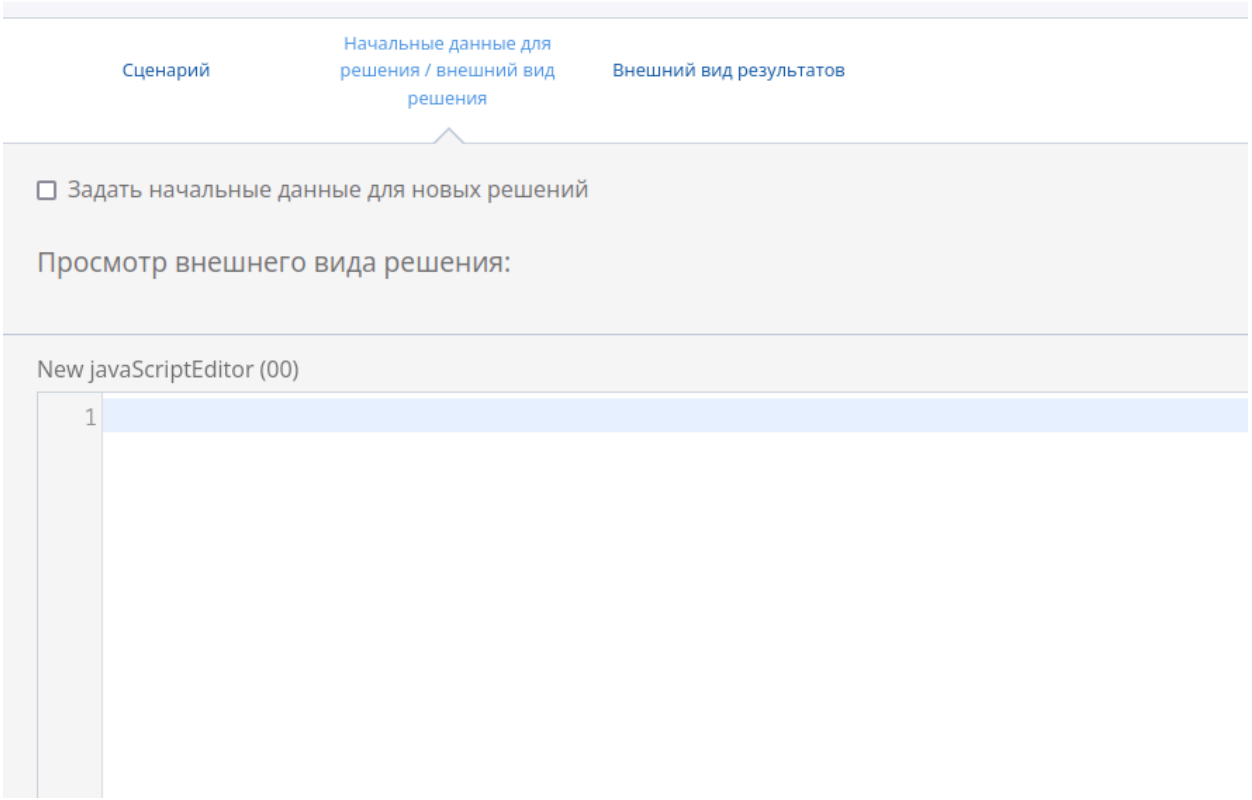
Для сценария типа “Симуляция” также необходимо задать прототип формы ввода данных в описательном формате JSON. В качестве решения может выступать как скрипт на языке JavaScript, так и поля для ввода параметров, с которыми будет происходить симуляция.

Если предполагается, что участники должны написать свою программу управления аппаратом, то в визуальном редакторе нужно добавить объект типа javaScriptEditor.

Основная статья: Редактор форм ввода

При редактировании свойств объекта можно указать имя (name) - заголовок, под которым будет располагаться поле ввода программы, - и используемое в программе имя (field). Оно понадобится для проверочного скрипта.

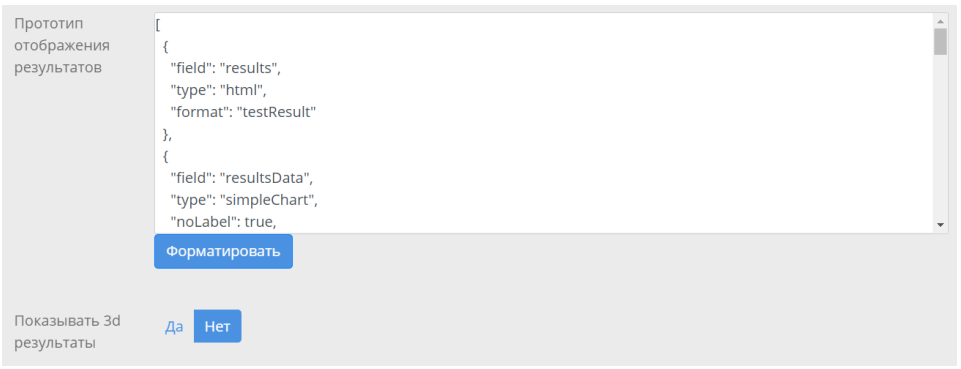
Перейдя во вкладку “внешний вид решения”, можно убедиться, что участник действительно сможет ввести программу в соответствующее поле.



Поле ввода программы на языке JavaScript также проверяет синтаксис: если допущены ошибки, то слева от нумерации строчек кода всплывёт уведомление.

Редактирование прототипа отображения результатов

После создания прототипа формы ввода данных нужно задать прототип отображения результатов. Следует настроить отображение результатов таким образом, чтобы участник не только узнал количество заработанных баллов и полученные достижения, но и смог посмотреть, что происходило во время симуляции с космическим аппаратом. Для этого можно настроить вывод в качестве результатов графиков тех величин, изменение которых непосредственно связано с решением.



Если выбрать “Показывать 3d результаты”, участник сможет посмотреть в отдельной вкладке траекторию движения спутника на орбите, узнать его скорость.

Прототип отображения результатов задаётся в формате JSON только в текстовом виде. Он также является массивом (в квадратных скобках) из объектов, каждый из которых имеет свои значения свойств, таких как “field”, “type” и т.п. Свойства объектов заключаются в фигурные скобки, свойства и объекты отделяются друг от друга запятыми.

Чтобы отобразить количество заработанных баллов, потребуется объект типа “html”:

```
[
  {
    "field": "results",
    "type": "html",
    "format": "testResult"
  },
<...>
]
```

Объект типа “simpleChart” позволяет строить графики зависимости некоторой величины Y от величины X. Например, чтобы построить график широты и долготы от времени, понадобится следующий код:

```
{
  "field": "resultsData",
  "type": "simpleChart",
  "noLabel": true,
  "name": "График широты, долготы",
  "xSeries": {
    "field": "t",
    "name": "Время",
    "unit": "c"
  },
  "ySeriesArr": [
    {
      "field": "s.0.m.p.la",
      "name": "Широта",
      "unit": "°"
    },
    {
      "field": "s.0.m.p.lo",
      "name": "Долгота",
      "unit": "°"
    }
```

```
 ]
}
{
"field": "resultsData", название переменной, содержащей результаты симуляции
"type": "simpleChart", тип, позволяющий строить график
"noLabel": true, название графика не отображается
"name": "График широты, долготы", название графика
"xSeries": { ось X
"field": "t", переменная времени
"name": "Время", названи оси
"unit": "c" единицы измерения величины по оси X
},
"ySeriesArr": [ ось Y, это массив
{
"field": "s.0.m.p.la", переменная широты
"name": "Широта", названи оси
"unit": "°" единицы измерения величины по оси Y
},
{
"field": "s.0.m.p.lo", переменная долготы
"name": "Долгота",
"unit": "°"
}
]
},
```

Некоторые имена переменных для свойств объектов “field”, которые можно использовать для получения информации (значений соответствующих им величин):

- t - время
- s.*.m.av.0 - угловая скорость аппарата по оси X
- s.*.m.av.1 - угловая скорость аппарата по оси Y
- s.*.m.av.2 - угловая скорость аппарата по оси Z
- s.*.m.p.al - высота аппарата над поверхностью
- s.*.m.p.la - широта аппарата
- s.*.m.p.lo - долгота аппарата
- s.*.m.o.0 - кватернион ориентации аппарата, элемент W
- s.*.m.o.1 - кватернион ориентации аппарата, элемент X
- s.*.m.o.2 - кватернион ориентации аппарата, элемент Y
- s.*.m.o.3 - кватернион ориентации аппарата, элемент Z

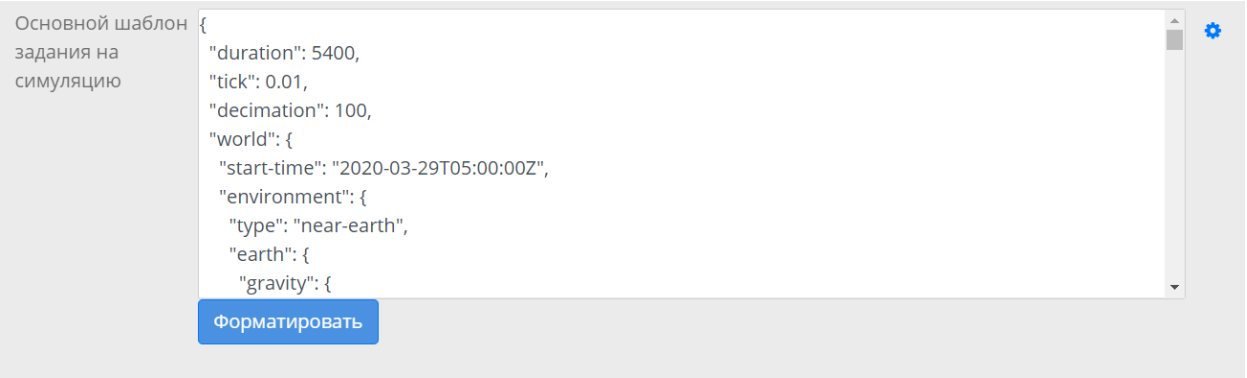
Вместо звёздочки в вышеперечисленных именах переменных должен стоять номер аппарата (если аппарат один, его номер - 0).

Чтобы отобразить трек аппарата на поверхности Земли, нужно добавить объект типа “simpleMap”. Долгота и широта аппарата будут отображаться на карте.

```
{
  "field": "resultsData",
  "type": "simpleMap",
  "noLabel": true,
  "name": "Траектория",
  "timeField": "t",
  "latFieldsArr": [
    "s.0.m.p.la"
  ],
  "lonFieldsArr": [
    "s.0.m.p.lo"
  ]
}
```

Редактирование основного шаблона задания на симуляцию

Основной шаблон задания на симуляцию в формате JSON можно задать как в текстовом виде, так и с помощью визуального редактора.



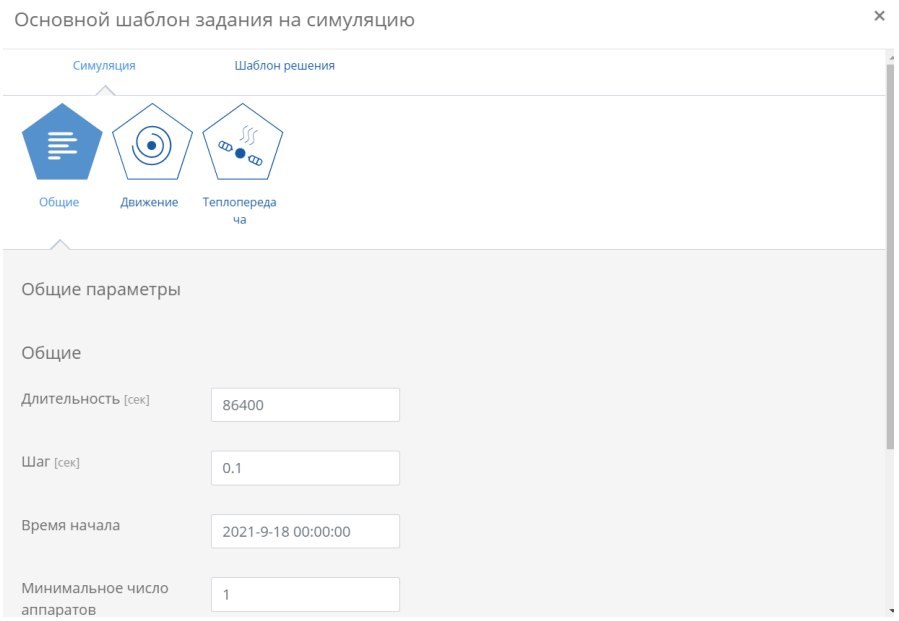
Удобнее воспользоваться визуальным редактором.

В шаблоне задания на симуляцию можно задать параметры симуляции, разделённые на три раздела: общие параметры, параметры движения и параметры теплопередачи и шаблон решения, где выбираются спутники и наземные станции. При создании сценария все эти параметры заданы стандартными значениями.

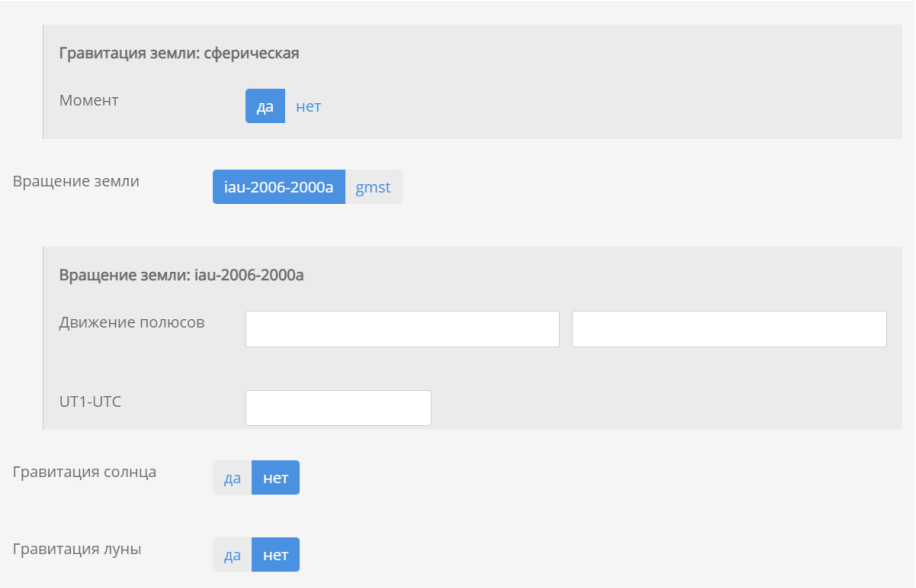
Во вкладке с общими параметрами симуляции нужно задать:

- Длительность симуляции, [сек];
- Шаг [сек] - время такта симуляции в секундах (норма 0,1 с);
- Абсолютные дата и время начала симуляции;

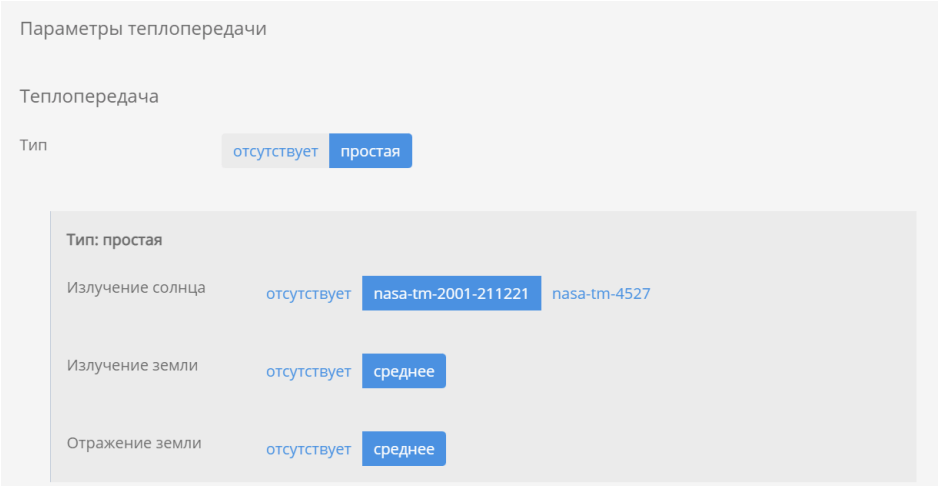
- Минимальное и максимальное число аппаратов;
- Максимальное и минимальное число станций.



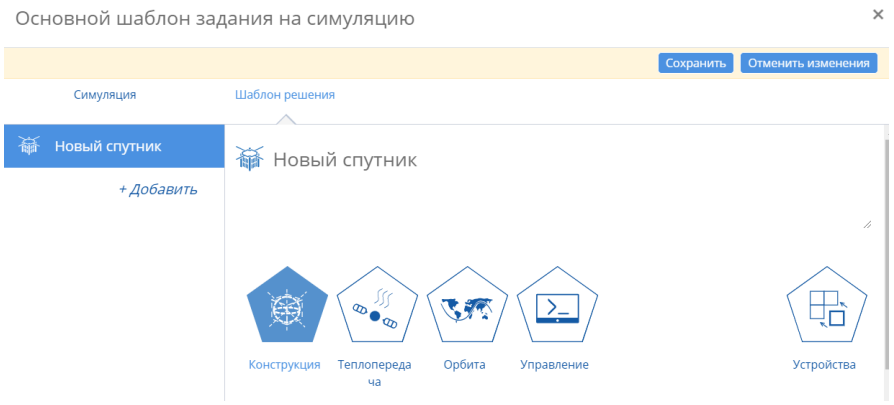
Параметры движения симуляции позволяют настроить, с какой точностью будет происходить расчёт и что будет учитываться при симуляции: считаем ли мы Землю сферой, как представляем её вращение, будем ли учитывать гравитацию Луны и Солнца.



Также можно задать параметры теплопередачи: излучение от Солнца, Земли, отражение Земли.

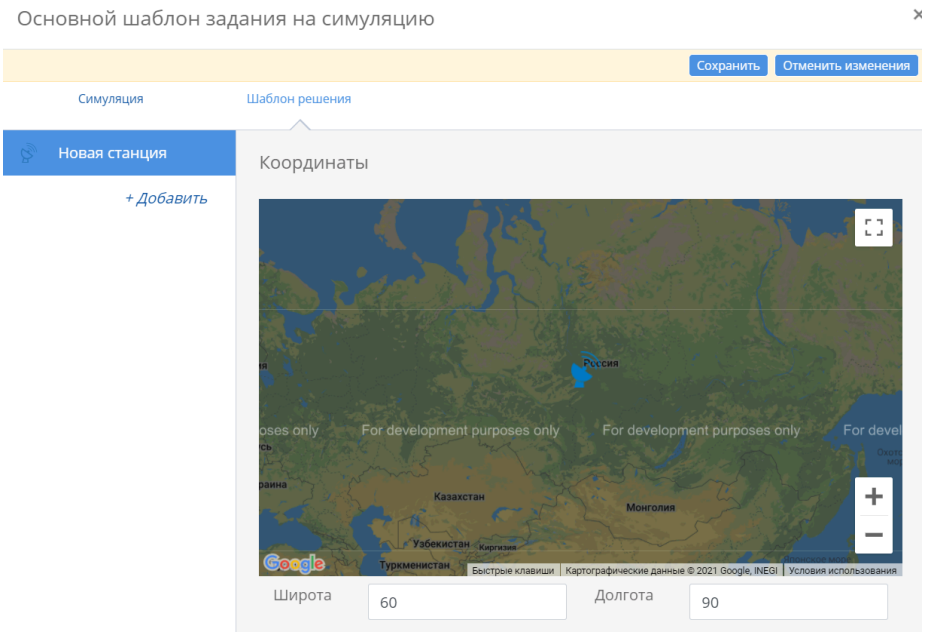


Во вкладке “Шаблон решения” нужно добавить как минимум один новый спутник - будет происходить симуляция именно его работы и движения на орбите.



Необходимо задать параметры спутника, а именно: его конструкцию, теплопередачу, данные о его движении на момент начала симуляции (параметры орбиты и ориентацию), состав и параметры устройств, расположенных на борту. Дополнительно во вкладке “Управления” можно задать программу управления спутником на языке JavaScript, которая будет выполняться дополнительно к программе, которую участник напишет в качестве решения.

Если предполагается взаимодействие спутника с наземными станциями, нужно добавить их и настроить их параметры. Требуется задать точные координаты станции, определить состав устройств, доступных к работе (антенна, приёмник, передатчик) и задать их характеристики, а также написать программу управления станцией в виде программы на языке JavaScript.

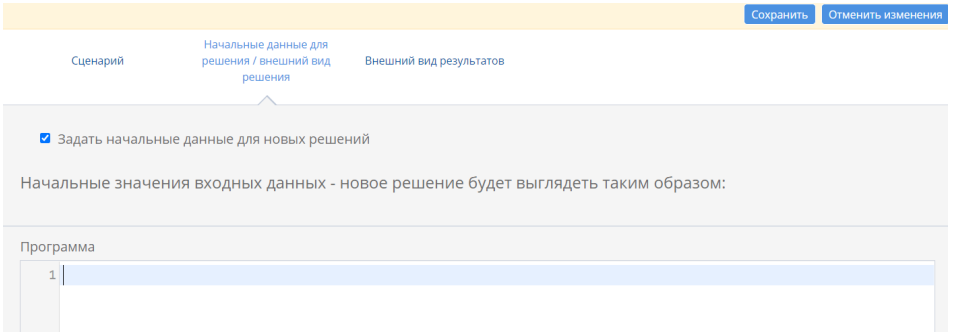


Оценка решения

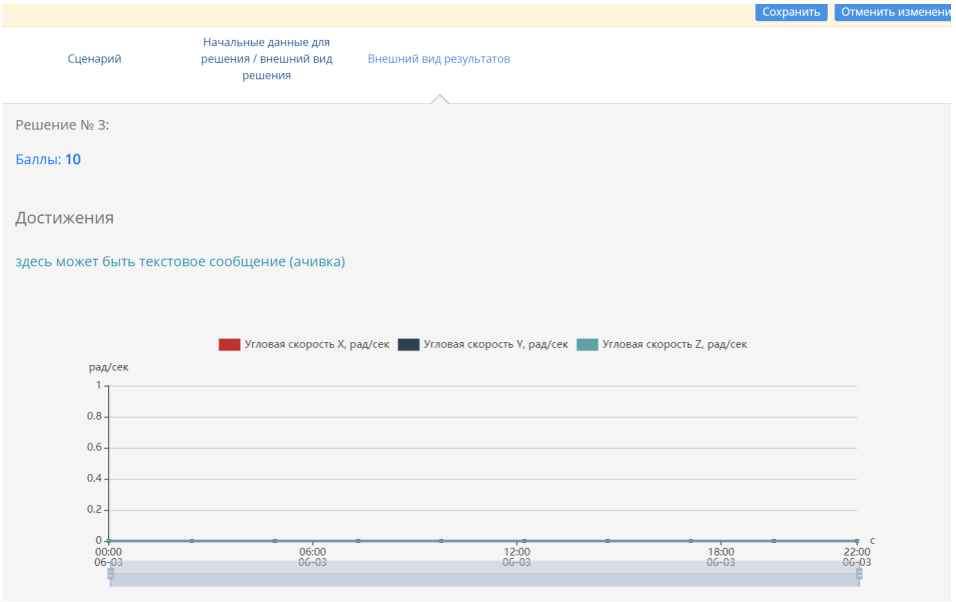
После завершения редактирования шаблона задания на симуляцию необходимо определить, как будет происходить оценка решения. Для этого в поле “Скрипт проверки (на JavaScript)” нужно поместить соответствующий скрипт. В нём должно происходить начисление баллов после проверки выполнения заданных условий.

Завершение редактирования сценария

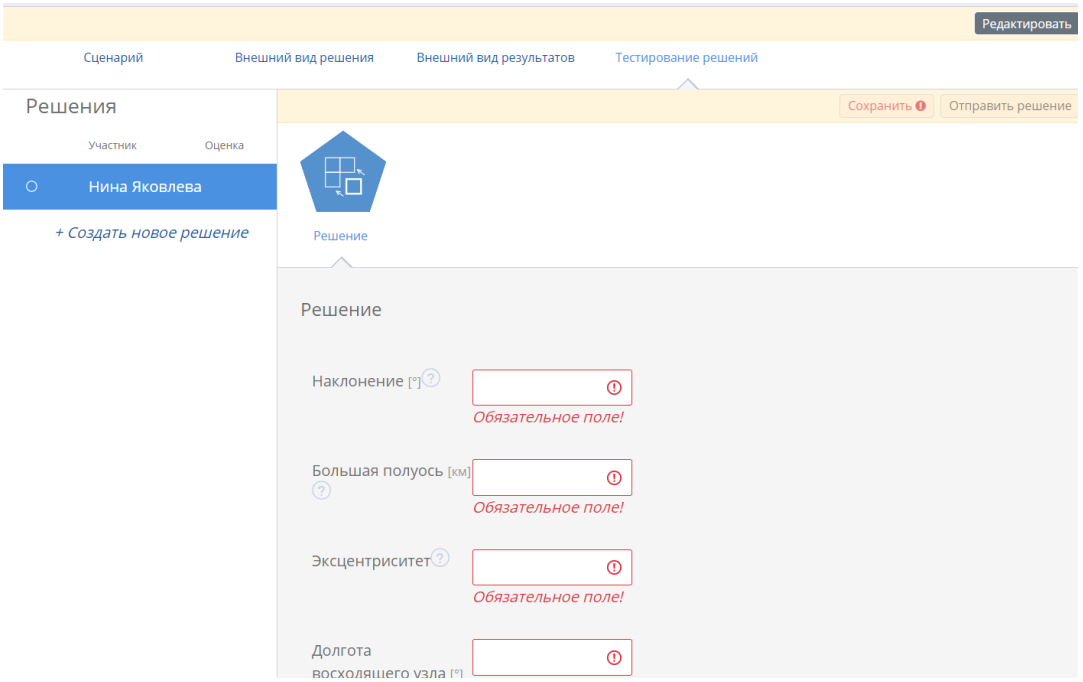
Кроме настройки сценария, можно также задать начальные данные для новых решений, осуществить просмотр внешнего вида решения и задать описание решения. Для этого потребуется написать скрипт на языке “JavaScript”.



Также можно посмотреть внешний вид результатов, зайдя в соответствующую вкладку.



Когда сценарий полностью настроен, нужно нажать кнопку “Сохранить”, после чего можно перейти во вкладку “Тестирование решений”, чтобы проверить работу сценария. Количество отправляемых решений для тестирования не ограничено.



После ввода решения нужно сначала нажать кнопку “Сохранить”, а затем - “Отправить решение”. Будет произведена симуляция и через некоторое время (оно зависит от длительности симуляции, количества шагов и её точности, определённых в шаблоне на задание симуляции) появятся результаты: количество баллов, полученные достижения, графики. Таким образом можно определить, соответствуют ли теоретические расчёты полученным результатам, верно ли начисляются баллы, не возникли ли какие-то непредвиденные ошибки.

4.2.7. Сценарий типа "Программа (Python)”

Данный сценарий позволяет проверить одну или несколько предложенных участников программ на Python.

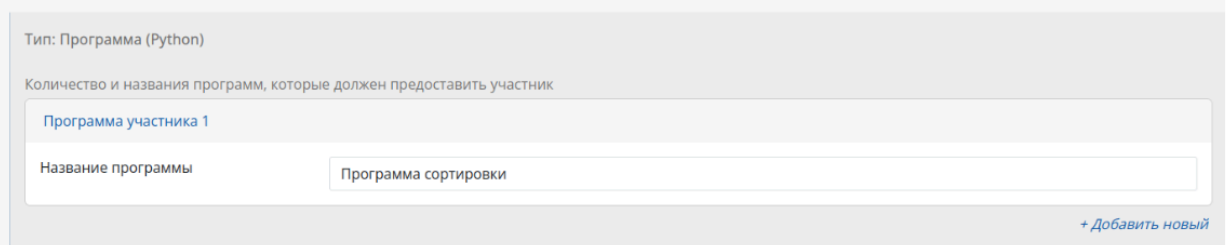
Для оценки можно использовать заданные фиксированные данные или генерировать случайные входные данные при каждом запуске.

Есть возможность запускать несколько разных программ участника, в том числе последовательно, с промежуточным контролем или изменением результатов (пример - задача на помехоустойчивое кодирование, в которой участник должен написать программы кодирования и декодирования, а в закодированный результат программой проверки могут быть случайным образом добавлены помехи.

Основные параметры сценария:

Количество и названия программ, которые должен предоставить участник

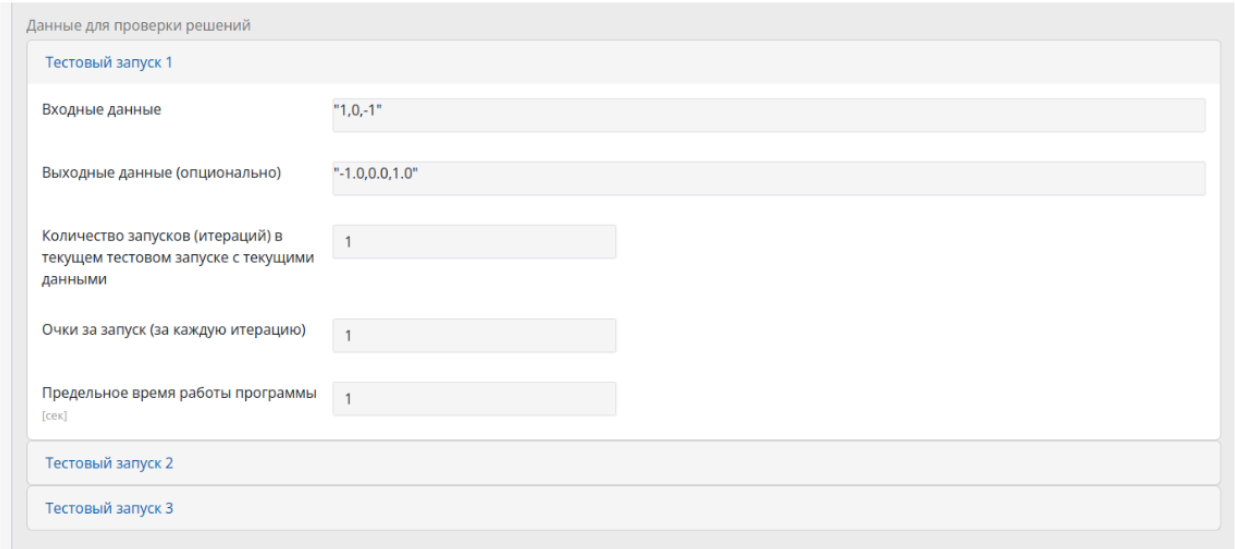
Указывается название для каждой программы, которую должен предоставить участник. В массиве пользовательских программ, доступном программе оценки, такие программы будут идти в приведенном здесь порядке.



Данные для проверки решений

Здесь можно добавить один или нескольких наборов данных для проверки решения участника.

- Входные данные - собственно тестовые данные в любом формате многострочного текста (простая строка, JSON и т.п.) Если ваша программа проверки не использует фиксированные входные данные, это значение будет проигнорировано.
- Выходные данные - данные, с которыми будет сравниваться результат программы участника (или любые иные дополнительные даные), если они нужны для вашей программы оценки
- Очки за запуск - сколько баллов начисляется при успегном прохождении программы запуска на этих данных
- Предельное время работы программы - максимум процессорного времени, после которого программа, если не завершится к тому времени, будет прекращена принудительно.



Параметры проверки решений

- Обработка результатов: суммировать или усреднять баллы за тесты
- Множитель для итоговых баллов: нормировать полученные баллы, домножив на это число, при необходимости

Скрипт проверки (на Python)

Единственная обязательная функция скрипта проверки это run(), именно она вызывается для каждого набора проверочных данных из списка выше.

Функция run должна возвращать структуру, в которой могут быть следующие поля:

Поле	Назначение	Примечание
score	Начисленные очки	Целое число, обязательное
message	Присвоенное "достижение"	Строка, не обязательное
log	Записи лога	Многострочный текст, не обязательное

Пример скрипта проверки с комментариями (в данном случае задача на сортировку чисел, входящие тестовые данные предоставлены в JSON формате):

```
# Подключение библиотек

# В данном случае мы работаем со стандартным вводом и выводом, а тестовые данные у нас в формате JSON

import os,sys,io,json

# Нам потребуется запускать пользовательские программы
```

```
import importlib

import importlib.machinery

import importlib.util


# Функция, формирующая запись для лога из вывода stdout

def get_log(user_exception_str = None):

    return '\\n'.join([s for s in [sys.stdout.getvalue(), user_exception_str] if s])


# Функция, формирующая в случае ошибки результат с начислением 0 баллов и соответствующим "достижением"

def return_user_error(user_error, user_exception_str = None):

    return {'score': 0, 'message': user_error, 'log': get_log(user_exception_str)}


# Основная функция запуска пользовательских программ, проверки и начисления очков, вызываемая для каждого теста

# Ключевые параметры

# input - данные для проверки решений (входящие)

# output - данные для проверки решений (исходящие)

# user_programs - массив программ участника в составе решения

def run(run_index, iteration_index, input, output, user_programs):


    # Берем входящие данные для проверки решений из теста

      # Также можно было взять исходящие данные из теста (в данном случае это не требуется)

       # Также можно было вместо этого сформировать случайные данные только для этого запуска

    input_data = json.loads(input)


    # Получаем код программы (в данном случае она одна, но их могло быть несколько)

    user_source = importlib.util.decode_source(user_programs[0])


    # Получаем параметры исполнения
```

```
        user_spec = importlib.machinery.ModuleSpec("user", None)
        user_module = importlib.util.module_from_spec(user_spec)

        # Подготавливаем потоки ввода и вывода, в поток ввода направляем входящие данные теста
        sys.stdin = io.StringIO()
        sys.stdin.write(input_data['message'])
        sys.stdin.seek(0)
        sys.stdout = io.StringIO()
        sys.stdout.close = sys.stdout.flush

        # Подготавливаем переменные для сохранения информации об ошибках и о начислении очков
        error = None
        score = 0

        # Запускаем пользовательскую программу, в случае ее нештатного завершения возвращаем ошибку
        try:
            exec(user_source, user_module.__dict__)
        except Exception as ex:
            return {'controller_error': 'System error: exec failed {}'.format(ex)}

        # Чистаем stdout  в ожидании реезультата работы пользовательской программы
        user_string = sys.stdout.getvalue()

        # На случай различного написания чисел для проверки формируем строку,
        # в которой все числа написаны и перечислены в том же порядке единообразно
        # если пользовательская строка не была перечислением чисел, то тест
        # все равно не пройден
        user_numbers = [float(num) for num in user_string.split(',')]
```

```
user_string = ','.join(map(str, user_numbers))


# Теперь получаем правильный ответ - сортируем строку из теста
# Вместо мы могли бы просто взять "Выходные данные" из теста
input_data['data'] = map(float, input_data['data'])
input_data['data'] = sorted(input_data['data'])
sorted_string = ','.join(map(str, input_data['data']))


  # Если пользовательское решение не совпадает с правильным ответом, возвращаем ошибку
# и 0 баллов с помощью функции, которую мы заготовили выше
if sorted_string != user_string:
   return return_user_error("[-] Test " + str((run_index + 1)) + " failed!")


# Подготавливаем полученные за тест очки и достижения (достижения не обязательны)
score = 1
message = "[+] Test " + str((run_index + 1)) + " passed!"


    # Возвращаем массив результата, включающий записи о полученных очках, достижениях и логе
return {'score': score, 'message': message, 'log': get_log()}
```

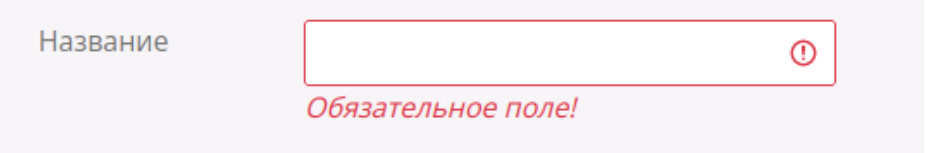
4.3. Создание и оформление материалов (общие сведения)

В этом разделе приведено описание некоторых общих для всей системы элементов управления и редактирования.

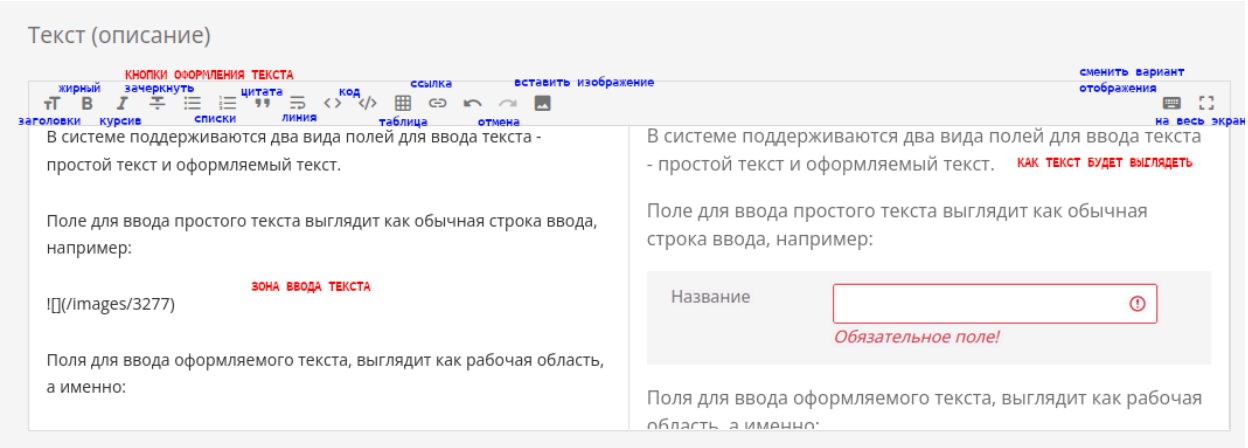
4.3.1. Создание и оформление текста

В системе поддерживаются два вида полей для ввода текста - простой текст и оформляемый (markdown) текст.

Поле для ввода простого текста выглядит как обычная строка ввода, например:



Поля для ввода оформляемого (markdown) текста, выглядит как рабочая область, а именно:



Такая рабочая область разделена на три зоны:

- Кнопки оформления текста
- Зона ввода текста
- Как текст будет выглядеть (формируется автоматически на основании введенного текста)

Все содержимое вводится в зону ввода текста. Оформление текста осуществляется добавлением специальных символов в стандарте markdown (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet - здесь можно посмотреть справку по этому стандарту).

Вам не обязательно знать и запоминать их, они сами проставляются при нажатии соответствующих кнопок оформления (см. изображение выше).

Примеры

Заголовки

Оформляется одним или нескольким знаком # перед текстом. Например так:

```
# Примеры

## Заголовки
```

Кнопками: выделите текст, не имеющий переносов строк и встроенного оформления. Нажмите первую кнопку в ряду и выберите тип заголовка.

Результат см. прямо в этом тексте, тут используются заголовки 1-го и 2-го уровня.

1) Жирный

Оформляется знаками ** справа и слева от текста. Результат выглядит вот так (введено: **вот так**).

2) Курсив

Оформляется знаками * справа и слева от текста. Результат выглядит *вот так* (введено: *вот так*).

3) Зачеркнуть.

Оформляется знаками ~~ справа и слева от текста. Результат выглядит вот так (введено: ~~вот так~~).

4) Немаркированный список

Оформляется знаками * , начинающими каждую новую строку, и отделенными от остальных слов пробелом. То есть, например так:

* Раз
* Два

Результат выглядит вот так:

- Раз
- Два

Если указать несколько астерисков через пробелы (например: * * Два), то уровень списка может быть повышен. Выглядит так:

- Раз
    - Два

5) Маркированный список

Также, но начинается с числа и точки (Например: 1.), начинающими каждую новую строку, и отделенными от остальных слов пробелом. То есть, например так:

1. Раз
2. Два

Результат выглядит вот так:

1. Раз
2. Два

6) Цитата

Начинается с символа “>” . Вводится так: “> Мой дядя, самых честных правил…”

Выглядит так: “Мой дядя, самых честных правил…”

7) Линия

Это просто. Пишем: —
Получаем:

---

8) Код

Это оформление предназначено для того, чтобы приводить в тексте примеры, которые должны показываться как есть (например код программ). Есть два типа оформления:

9) Код в строке.

Оформляется так: Тут нужно написать такой `текст`
Выглядит так:
Тут нужно написать такой текст
Код на нескольких строчках. Оформляется так:

```
#include<stdio.h>
#include<introsat.h>
```

Выглядит так:
#include<stdio.h>

#include<introsat.h>

10) Таблица

Тут несколько сложнее. Столбцы должны быть разделены знакамии | и пробелами. Каждая строка текста - строка таблицы. У таблицы должен быть заголовок, отделенный строкой, содержание каждой ячейки которой - три дефиса (---)

То есть вводить нужно так:

| Фамилия | Дней работы |

| --- | --- |

| Иванов | 3 |

| Петров | 1 |

| Сидоров | 5 |

А выглядеть будет так:

Фамилия	Дней работы
Иванов	3
Петров	1
Сидоров	5

11) Гиперссылка

Сначала в квадратных скобках указывается название ссылки. Затем сразу, без пробела, в кругулых скобках - адрес. Например так: [Конструктор Интросат](https://introsat.ru)

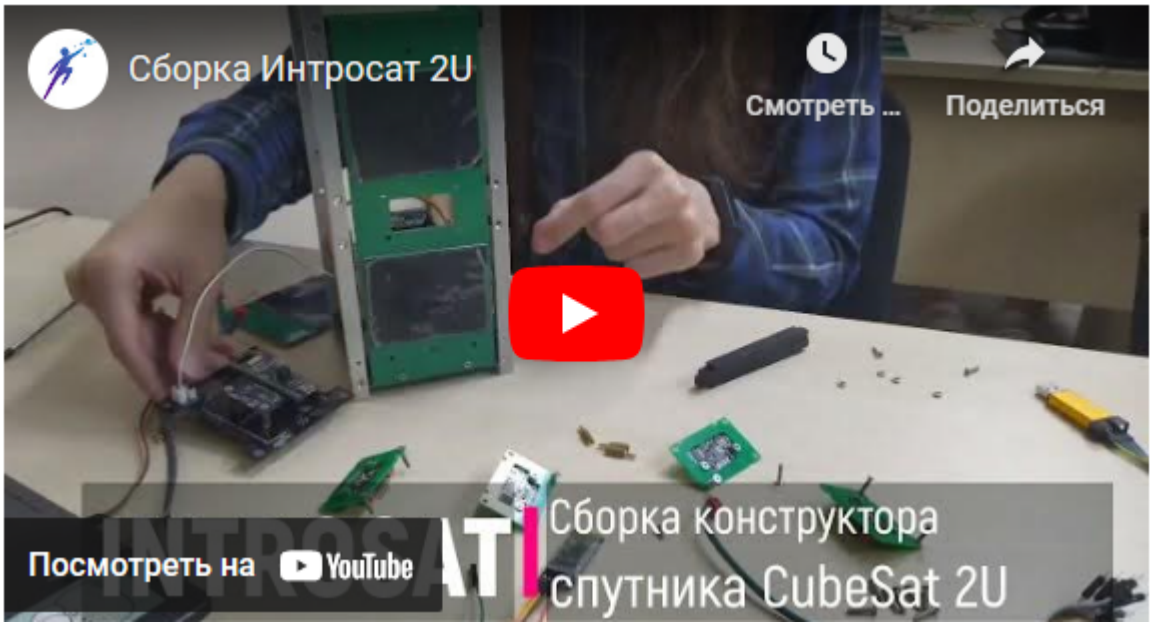
Результат выглядит так: Конструктор Интросат

Если нужно вставить предпросмотр ссылки, то перед квадратными скобками ставим восклицательный знак. ЭТО СПОСОБ ИНТЕГРИРОВАТЬ ВИДЕО, например:

Вставляем:

```
![](https://youtu.be/gpKlmO6aWwE)
```

Получаем:



12) Кнопки "Отменить" и "Вернуть"

Соответственно отменяют последнее действие или возвращают последнее отмененное действие.

13) Вставка и загрузка картинок

Код для вставки картинки: , где вместо троеточия - адрес картинки в интернете или в Орбите.

Обычно знать адрес картинки не нужно. Просто нажмите кнопку "Вставить картинку" и выберете, откуда ее вставить.

Для иллюстраций в тексте, например формул, графиков и т.п. рекомендуется добавлять новый файл, загрузив его с Вашего компьютера. Но можно добавить изображение из интернета или из каталога Орбиты.

Подробнее о добавлении изображений см. следующий раздел "Добавление изображений".

Переключение отображения

Изменить вариант отображения рабочей области:

- Зона ввода текста и зона отображения
- Только зона ввода
- Только зона отображения

На весь экран

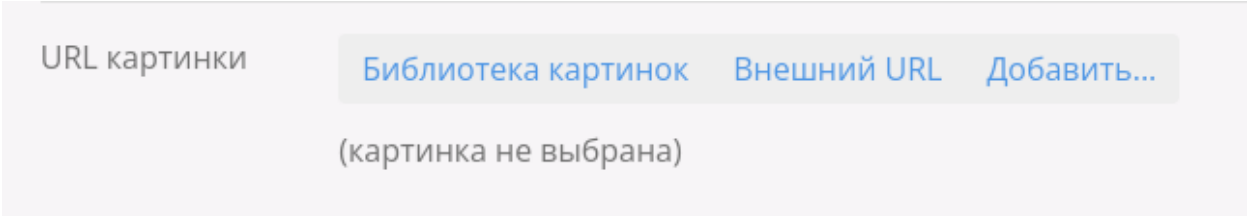
Развернуть/свернуть область ввода текста на весь экран.

4.3.2. Добавление изображений

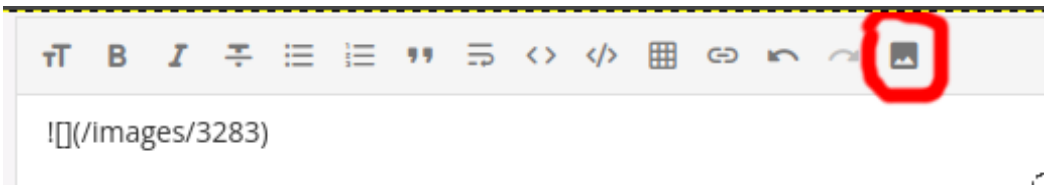
Вы можете загружать изображения на обложки событий, задач и курсов, а также вставлять изображения в тексты заданий, описаний, статей и так далее.

**Форма вставки изображения появляется **

1. В карточке события, сценария, страницы - При нажатии кнопок управления изображения. Например:

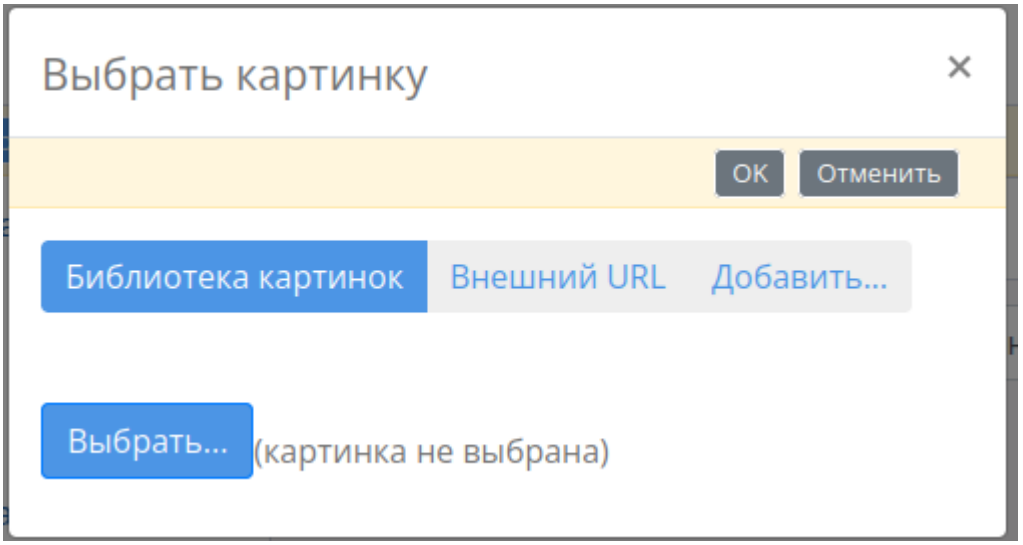


2. В тексте - при нажатии на кнопку "Вставить изображение". Если курсор стоит там, где еще нет ссылки на изображение - оно будет добавлено. Если он стоит на ссылке на изображение - оно будет изменено. Например:

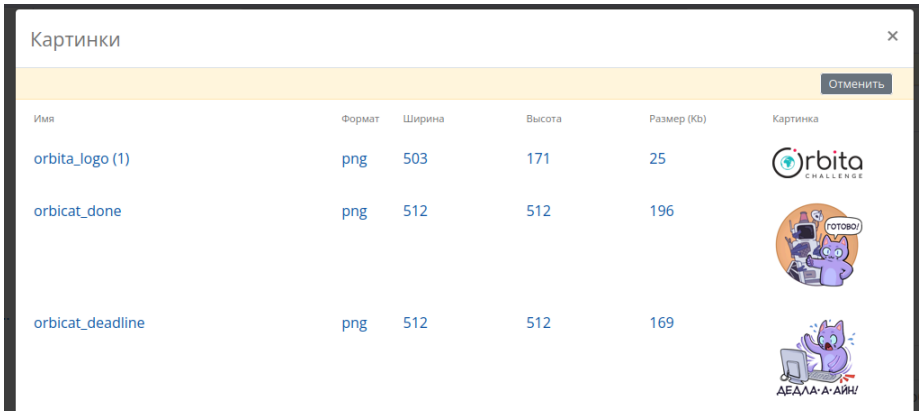


Диалог позволяет выбрать один из трех способов.

1. Выбор изображения из каталога. Нужно выбрать вариант "Библиотека картинок"

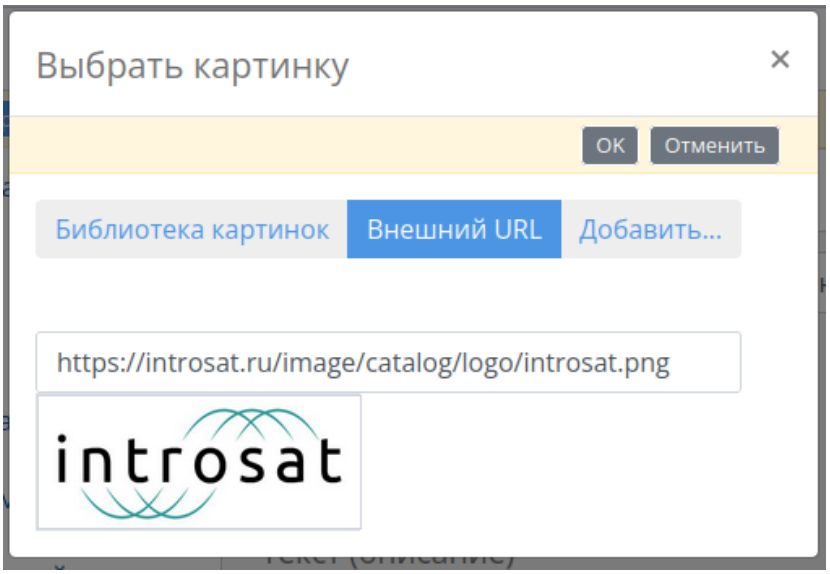


В этом случае выбирается изображение из каталога Орбиты или данной лицензии, например:



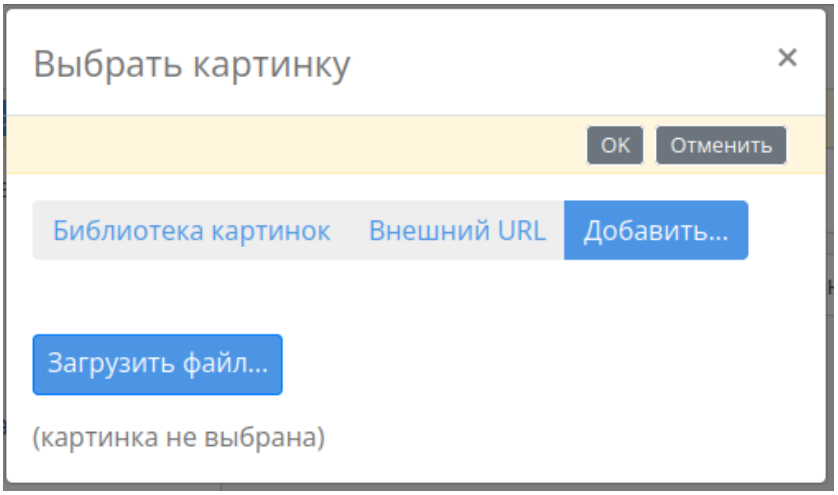
2. Выбор изображения из сети

Нужно выбрать "Внешний URL" и указать адрес картинки:



3. Загрузить с вашего компьютера

Нужно выбрать кнопку "Добавить", затем "Загрузить файл", выбрать файл и подтвердить выбор.



Внимание: при добавлении изображения с Вашего компьютера, оно тратит объем данных лицензии.

Внимание: изображения, добавленные с Вашего компьютера в текст страницы, будет удалены из Орбиты (и не будут более тратить объем данных лицензии), если будет удалена эта страница или это с нее будет удалена ссылка на это изображение.

5. Справка по API

5.1. Классы и основы объектно-ориентированного программирования

Перед решением задач участникам предлагается ознакомиться с API аппарата.

Аппарат в сервисе семейства "Орбита" состоит из т.н. устройств, представляющих единое целое с точки зрения проекта. Это может быть как отдельный датчик, так и целая подсистема. Устройство может поддерживать одну или несколько предопределенных функций.

При создании аппарата вы можете добавить к любым устройствам любые функции. Например, вы можете оснастить спутник средствами связи. Для этого можно:

1. Добавить устройство "Радиомодуль" с тремя функциями - "Приемник", "Передатчик" и "Антенна", или
2. Сделать два разных устройства с двумя функциями каждое, одно - только для приема, другое только для передачи, и у каждого будет своя антенна;
3. Сделать устройство с несколькими функциями "Передатчик", если оно может независимо работать на нескольких разных частотах
4. Добавить в устройство "Научный модуль" функцию "Передатчик", если по задумке разработчика это оборудование имеет свой радиомодуль и антенну.
5. И т.п.

Для более детального понимания, как устроено API, следует ознакомиться с принципами объектно-ориентированного программирования.

В рамках этого подхода среди прочего объявляются так называемые типовые классы объектов, обладающими собственными свойствами (подчиненными переменными и объектами) и функциями. Некоторые из этих свойств и функций в программе могут быть доступны только для чтения, другие позволяют управлять этим объектом через вызовы его функций с определенными параметрами.

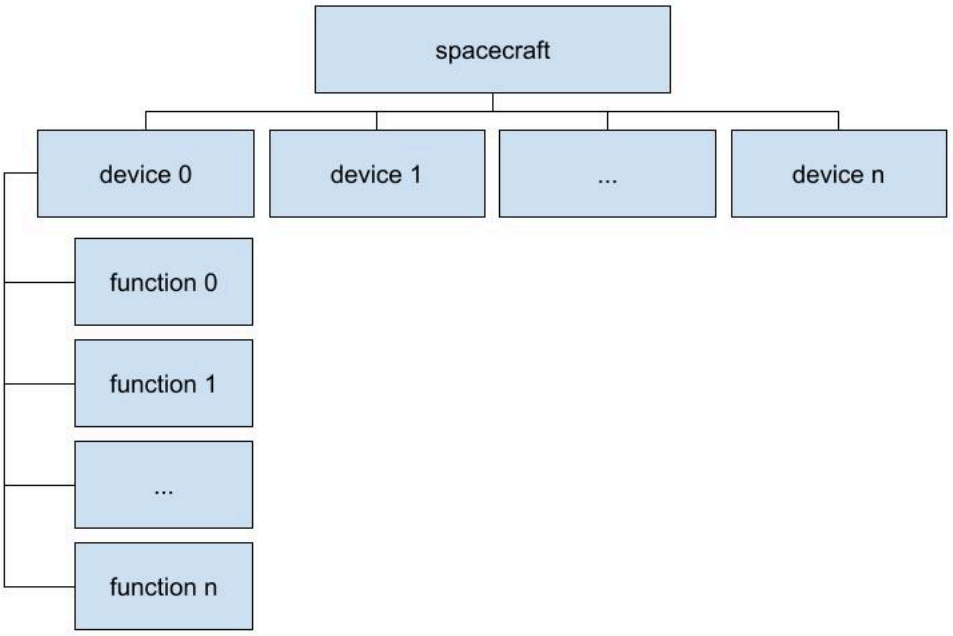
В сервисах семейства "Orbita" космический аппарат представлен глобальным объектом класса "Spacecraft", который хранит все параметры аппарата, его бортовые устройства, а также имеет некоторые типовые функции, такие как отсчет времени от момента старта миссии.

В частности, в состав аппарата и/или нахземной станции входит набор бортовых устройств, представленных объектами класса "Device", перечисленных в неизменяемом массиве "Spacecraft.Devices".

У каждого устройства в аппарате есть порядковый номер (начинается с 0). Кроме того, у него может быть уникальное название, задаваемое при конструировании аппарата. Обращение к устройству обычно осуществляется через этот порядковый номер, например:

```
accumulator = spacecraft.devices[0];
```

Каждое устройство может выполнять одну или несколько функций, определяющих его роль в аппарате (например, аккумулятор или радиопередатчик) и представленных объектами класса Function и классов-наследников (Accumulator, Heater, ...). У каждой функции в устройстве также есть порядковый номер (также начинается с 0) и может быть название.



На старте миссии конструкция аппарата считается заданной.

Пример массива устройств аппарата

Номер	Тип устройства	Включено в начале симуляции	Функции устройства и их порядковые номера	Примечание
0	Аккумулятор	True	Аккумулятор (0)	Может быть заряжен изначально, накапливает заряд при избытке питания и отдает при недостатке

1	Система солнечных батарей	True	Солнечная панель (0), Солнечная панель (1), Солнечная панель (2), Солнечная панель (3), Солнечная панель (4), Солнечная панель (5)	Набор из нескольких солнечных панелей, ориентированных на 6 разных гранях аппарата, обеспечивают питание аппарата, если на них падает солнечный свет
2	Маневровый двигатель	False	thruster (0), propellant-tank(1)	Небольшой бортовой реактивный двигатель для смены орбиты. Содержит реактивное устройство и "топливный" бак (например холодный газ под давлением)

Пример устройства и значение его функций в начальный момент симуляции

Устройство: маневровый двигатель

Номер функции	Тип функции	Включена в начале	Значение при обращении	Комментарий
0	thruster	False	0	Функция устанавливает и возвращает тягу. На начало симуляции двигатель доступен, но вывод рабочего тела не осуществляется.
1	propellant-tank	True	0.572	Запас рабочего тела

5.2. Написание программы управления на языке JavaScript

Ниже приведена краткая справка по написанию программы на языке JavaScript, достаточная для написания простого кога управления аппаратом.

Общие положения

Первой строкой программы является директива, позволяющая использовать более строгий вариант JavaScript:

'use strict';

После каждой инструкции (команды) должна ставиться точка с запятой.

Переменные объявляются с помощью ключевых слов var или let. Чтобы переменную можно было использовать во всех функциях скрипта, можно сделать её глобальной: в таком случае её необходимо объявить в начале скрипта, до всех функций.

var thruster;

Инициализировать переменную, то есть, сохранить в неё какое-то конкретное значение, можно с помощью операции присваивания - знака

равенства (не путать с двойным знаком равенства - этот оператор позволяет сравнивать выражения).

let time = 0;

Для условного ветвления существует инструкция if, к которой можно также добавить блоки else if и else. Проверяемые условия записываются в круглых скобках после слова if. Если выражение, записанное в условии, истинное, то выполняется соответствующий блок.

Инструкции, которые должны быть выполнены, берутся в фигурные скобки.

```
if (time == 10) {
// делать что-то
} else if (time > 15) {
// делать что-то ещё
} else {
// делать что-то совсем другое
}
```

Доступ к свойствам объекта осуществляется через операцию точка, которая ставится после имени интересующего объекта.

Обратимся к свойствам класса Spacecraft: если мы хотим узнать время, прошедшее с начала полёта, необходимо сначала записать имя конкретного объекта - spacecraft (это спутник, для которого вы пишете программу), - затем точку, после чего обратиться к соответствующему свойству - flight_time. Можно сразу сохранить значение времени в объявленную ранее переменную time:

time = spacecraft.flight_time;

Класс Spacecraft содержит в себе несколько объектов класса Device, имя которых записывается как devices[i], где i - номер устройства в таблице, данной выше. Внутри каждого объекта класса Device так же есть один или несколько объектов класса Function, к которым можно обратиться по имени functions[j], где j - номер функции в таблице. Для каждого объекта класса Device существуют свойства, приведённые в списке выше. Допустим, мы хотим узнать угловую скорость по направлению Y с помощью гироскопа и сохранить её в некоторую переменную Vy.

Список возможного оборудования и его функций можно посмотреть в следующем разделе.

Состав устройств конкретного аппарата, порядок его устройств и функций или задан в условиях сценария, или задается пользователем самостоятельно в режиме свободного проектирования.

Допустим, среди бортовых устройств есть Гироскоп (его номер 2) и среди его функций есть датчик угловой скорости по оси X (его номер 0).

Для получения значения угловой скорости в текущий момент обращаемся сперва к аппарату - spacecraft, затем к устройству - devices[2], затем к функции - functions[0], и в последнюю очередь - к самому свойству для функции гироскопа - angular_velocity.

var Vy = spacecraft.devices[2].functions[0].angular_velocity;

Чтобы не писать каждый раз такие громоздкие конструкции, можно создавать специальные переменные (лучше всего, с говорящим названием) и затем обращаться к требуемым свойствам через них.

Например, чтобы изменить тягу двигателя (свойство thrust класса “Двигатель” функцией “Управление тягой”), можно сначала определить переменную thruster, которая сразу позволит обращаться к необходимой функции, а затем уже менять свойство thrust.

var thruster = spacecraft.devices[0].functions[0];

После чего зададим двигателю максимальную тягу с помощью свойства maximum_thrust:

var thruster = spacecraft.devices[0].functions[0];
thruster.thrust = thruster.maximum_thrust;

Структура программы управления

Глобальные переменные и директивы объявляются в начале программы. Если необходимо выполнить какой-либо код один раз в начале симуляции, следует поместить его в функцию setup().

Основной код управления помещяется внутри функции loop(), которая вызывается циклически все время симуляции. Во избежание прерывания функции по таймеру с ошибкой следует избегать длительного исполнения кода в этой функции.

Ниже приведён пример простой программы, где для удобства создаётся переменная для обращения к функции “Управление тягой” объекта класса “Двигатель”. Двигатель включен в начале симуляции, но начиная с сотой миллисекунды полета должна быть задана тяга, равная нулю, - то есть, двигатель можно считать выключенным.

```
'use strict'; // Объявлена директива
var thruster; // Объявлена переменная для удобства работы с устройством "Двигатель"
```

```
        // В начале симуляции устанавливаем, что "thruster" указывает на
    первую функцию первого устройства.
    function setup() {
        thruster = spacecraft.devices[0].functions[0];
    }
    // Отключаем тягу начиная с сотой миллисекунды
    function loop() {
     if (spacecraft.flight_time >= 100) {
     thruster.thrust = 0;
        }
    }
```

5.3. Обзор устройств и функций космического аппарата

Ниже приведены свойства и методы, которые необходимо использовать при написании бортовой программы управления космическим аппаратом.

Напоминаем, что минимальная управляемая единица оборудования - это функция. При создании аппарата вы можете добавить к любым устройствам любые функции. Например, вы можете оснастить спутник средствами связи. Для этого можно:

1. Добавить устройство "Радиомодуль" с тремя функциями - "Приемник", "Передатчик" и "Антенна", или
2. Сделать два разных устройства с двумя функциями каждое, одно - только для приема, другое только для передачи, и у каждого будет своя антенна;
3. Сделать устройство с несколькими функциями "Передатчик", если оно может независимо работать на нескольких разных частотах
4. Добавить в устройство "Научный модуль" функцию "Передатчик", если по задумке разработчика это оборудование имеет свой радиомодуль и антенну.
5. И т.п.

Свойства класса Spacecraft

Свойство	Значение
flight_time	Значение времени от начала полета в секундах (с точностью до миллисекунд).
calendar_utc_time	Глобальное время в симуляции по UTC

devices	Массив, содержащий устройства (объекты класса Device).

Общие свойства класса Device

Свойство	Значение
enabled	Состояние устройства (true - включено, false - выключено)
failed	Признак того, что устройство вышло из строя (true - вышло из строя, false - работоспособно).
functions	Массив, содержащий функции (объекты класса Function и классов-наследников)

Общие методы класса Device

Метод	Описание
enable()	Включить устройство
disable()	Выключить устройство

5.4. Свойства класса Spacecraft для объективного мониторинга

Приведенные ниже свойства доступны для класса Spacecraft в скрипте автоматической оценки результата

Свойство	Описание
motion	Структура данных о положении, ориентации и скорости аппарата
thermodynamics	Структура данных о тепловых параметрах аппарата

electrics	Структура данных о состоянии системы электропитания аппарата

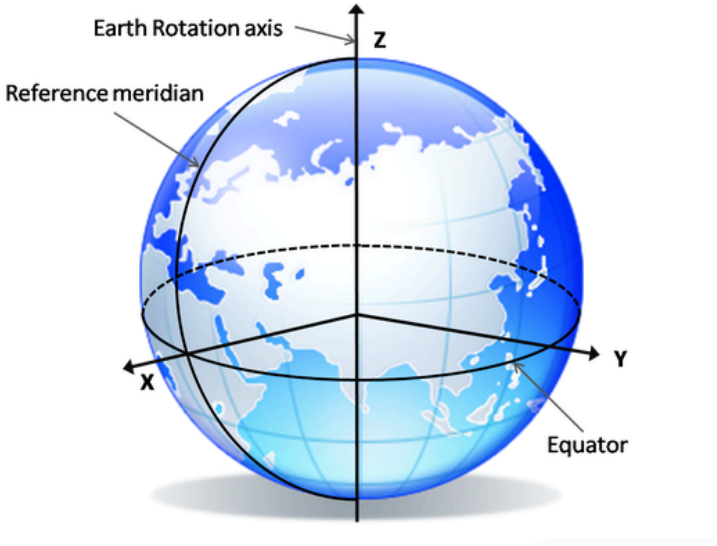
Структура "Motion"

Свойство	**Описание**
angular_velocity	Массив из трех значений. Угловая скорость по трем осям.
position	Массив из трех значений. Абсолютные координаты в неинерциальной системе координат (см ниже).
linear_velocity	Массив из трех значений. Линейная скорость по трем осям
orientation	Массив из четырех значений. Кватерниор текущей ориентации аппарата
location	Массив из трех значений. Пространственное положение в системе глобальной навигации (широта в градусах, долгота в градусах, высота над уровнем моря в метрах)

Примечание по неинерциальной системе координат.

В качестве системы координат, связанной с Землей используется земная система, для которой в симуляторе справедливо следующее:

1. Центр координатной системы лежит в центре масс Земли.
2. Ось z совпадает с осью вращения Земли и проходит через северный полюс.
3. Ось x проходит через пересечение нулевого меридиана и экватора.
4. Ось y дополняет систему координат до правой.


Earth Rotation axis
Z
Reference meridian
Y
X
Equator

Структура "Thermodynamics"

Свойство	Описание
temperature	Средняя температура аппарата

Структура "Electrics"

Свойство	Описание
accumulator_charge	Суммарный текущий заряд всех бортовых батарей в Джоулях
accumulator_capacity	Суммарная емкость всех бортовых аккумуляторов
power_production	Суммарное текущее поступление питания
power_consumption	Суммарный текущий расход питания
power_balance	Текущий энергобаланс в моменте

5.5. Работа с информацией и осуществление радиосвязи

В образовательной версии специфика получения и передачи в полезных данных задается администратором события.

Для этого у каждой из функций "Источник данных", "Потребитель данных", "Передатчик" и "Приемник" существует доступные только в скрипте оценки буфер данных и специальные методы по работе с ним:

Метод	Описание
push_queue(i,a)	Добавляет массив a типа Uint8Array к буферу устройства. i - индекс буфера (см. примечания)

pull_queue(i,m)	Забирает массив типа Uint8Array из буфера устройства и возвращает его. i - индекс буфера. m - максимальное количество байт, забираемых из буфера

Примечания:

1. У функций может быть несколько буферов, которые вы можете использовать в своих целях. Однако методы, доступные пользователям, работают только с буфером с индексом "0". Значение по умолчанию: 0.
2. Шаг симуляции задается в параметрах симуляции. Чаще всего это 0.1 секунды. Физическаяя скорость передачи равна количеству байтов за шаг, деленная на длительность шага. Так, при m = 2 корость будет 20 байтов в секунду.
3. Если в буфере меньше, чем m байт, вернётся всё что есть в буфере.

Общие принципы работы с данными:

1. Для производства потока нужных данных, автор события может добавлять такие данные на каждом шаге симуляции в буфер любого из используемых в симуляции устройств с функцией Источника данных.
2. Для передачи данных по радиосвязи на каждом шаге симуляции в скрипте оценки необходимо забирать данные с нужных Передатчиков одних аппаратов и помещать их в подходящие Приемники других аппаратов или наземных Станций. Объем передавыемых на каждом шаге данных (обычно это 0.1 секунды) определяется разработчиком задачи исходя из своих целей и технических параметров устройств.
3. Проверка радиовидимости и внесение помех могут осуществляться (или не осуществляться) исходя из дизайна задачи.
4. Обычно для контроля переданной информации используется чтение из буфера устройства "Приемник данных", однако использование этого устройства не является необходимым. Разработчик задачи в скрипте оценки может контролировать информацию на любом этапе, например считывая ее из целевого устройства типа "Приемник".

Ниже приведен пример кода скрипта оценки с активной работой с передатчиками, проверками видимости и повреждением передаваемой информации.

```
'use strict';

let MAX_HEIGHT = 3000000.0;
let PACKET_SIZE = 100;
let MAX_CHUNK_SIZE = Math.round(3600 * runtime.tick);
let PREAMBLE = [0xde, 0xad, 0xbe, 0xef];
```

```
let produced_storage = new Map();
let produced_index = -1;
let consumed_queue = [];

let score = 0.0;

function radians(angle) {
    return angle * Math.PI / 180;
}

function make_coords(latitude, longitude) {
    return {latitude: radians(latitude), longitude: radians(longitude)};
}

function calculate_elevation(station_location, spacecraft_location) {
    let height = spacecraft_location[2];
    let radius = world.environment.earth.radius;
    let coords_1 = make_coords(station_location[0], station_location[1]);
    let coords_2 = make_coords(spacecraft_location[0], spacecraft_location[1]);
    let cos_angle = Math.cos(coords_1.latitude) * Math.cos(coords_2.latitude);
    cos_angle *= Math.cos(coords_1.longitude - coords_2.longitude);
    cos_angle += Math.sin(coords_1.latitude) * Math.sin(coords_2.latitude);
    let sin_angle = Math.sqrt(1.0 - cos_angle * cos_angle);
    let x = radius * sin_angle;
    let y = radius * (1.0 - cos_angle) + height;
    return [Math.atan2(y, x) - Math.acos(cos_angle), Math.sqrt(x * x + y * y)];
}

function check_visibility(station_location, spacecraft_location) {
    let height = spacecraft_location[2];
    if (height > MAX_HEIGHT) {
        return false;
    }

    let elevation = calculate_elevation(station_location,spacecraft_location);
    if (elevation[0] < Math.PI / 9) {
        return false;
    }

    return true;
}

function setup() {
    runtime.add_score("default", 0.0);
}

function loop() {
    let produced_packet = null;
```

```
let flight_time = world.flight_time;
if (Math.floor(flight_time) > produced_index) {
    produced_index = Math.floor(flight_time);

    produced_packet = new Uint8Array(PACKET_SIZE);
    produced_packet.set(PREAMBLE, 0);
    produced_packet.set([(produced_index >> 0) & 0xff,
                         (produced_index >> 8) & 0xff,
                         (produced_index >> 16) & 0xff,
                         (produced_index >> 24) & 0xff], 4);
    produced_packet.set(runtime.random(92), 8);

    produced_storage.set(produced_index, Array.from(produced_packet));
}

let chunk_in_flight = null;

let spacecraft = world.spacecrafts[0];
if (spacecraft.active) {
    let producer = spacecraft.devices[1].functions[0];
    if (producer.active && (produced_packet !== null)) {
        producer.push_queue(0, produced_packet);
    }

    let transmitter = spacecraft.devices[0].functions[0];
    if (transmitter.active) {
        chunk_in_flight = transmitter.pull_queue(0, MAX_CHUNK_SIZE);
    }
}

if (chunk_in_flight !== null) {
    for (let i = 0; i < chunk_in_flight.length; i++) {
        let rolls = runtime.random(8);
        for (let j = 0; j < 8; j++) {
            if (rolls[j] < 3) {
                chunk_in_flight[i] ^= 1 << j;
            }
        }
    }
}

let consumed_chunk = null;

let station = world.stations[0];
if (station.active) {
    let visible = false;
    if (spacecraft.active) {
```

```
        visible = check_visibility(station.motion.location,
                                   spacecraft.motion.location);
    }

    let receiver = station.devices[0].functions[0];
    if (receiver.active && visible && (chunk_in_flight !== null)) {
        receiver.push_queue(0, chunk_in_flight);
    }

    let consumer = station.devices[1].functions[0];
    if (consumer.active) {
        consumed_chunk = consumer.pull_queue(0, MAX_CHUNK_SIZE);
    }
}

if (consumed_chunk !== null) {
    consumed_queue.push(...consumed_chunk);
    while (consumed_queue.length >= PACKET_SIZE) {
        if ((consumed_queue[0] != PREAMBLE[0]) ||
            (consumed_queue[1] != PREAMBLE[1]) ||
            (consumed_queue[2] != PREAMBLE[2]) ||
            (consumed_queue[3] != PREAMBLE[3])) {
            consumed_queue.shift();
            continue;
        }

        let consumed_index = (consumed_queue[4] << 0) +
                             (consumed_queue[5] << 8) +
                             (consumed_queue[6] << 16) +
                             (consumed_queue[7] << 24);
        if (!produced_storage.has(consumed_index)) {
            consumed_queue.splice(0, 4);
            continue;
        }

        let produced_content = produced_storage.get(consumed_index);
        let consumed_content = consumed_queue.slice(0, PACKET_SIZE);

        if (consumed_content.toString() != produced_content.toString()) {
            consumed_queue.splice(0, 4);
            continue;
        }

        score += 0.00048828125;
        if (score <= 30.0) {
            runtime.add_score("default", 0.00048828125);
        }
```

```
            produced_storage.delete(consumed_index);
            consumed_queue.splice(0, PACKET_SIZE);
        }
    }
}
```

5.6. Написание скрипта автоматической оценки

Скрипт автоматической оценки решения пишется на языке JavaScript и основан на тех же принципах, что и программа управления, со следующими особенностями:

1. В скрипте оценки одновременно есть доступ ко всем моделируемым объектам, в том числе ко всем аппаратам и всем наземным станциям симуляции. Доступ у свойствам и методам каждого аппарата и его бортового оборудования осуществляется через свойства и методы вселенной world.
2. В скрипте оценки доступны специальные функции начисления оценки, вывода логов и достижений, перечисленные в разделах ниже. Это осуществляется с помощью свойств и методов среды окружения runtime.
3. Для аппаратов и станций доступны дополнительные свойства объективного мониторинга из раздела выше.

Свойства вселенной world

Свойство	Описание
flight_time	Время, прошедшее с начала симуляции
spacecrafts	Массив спутников в симуляции
stations	Массив наземных станций в симуляции

Методы среды исполнения

Метод	Описание
add_score(t, x)	Добавить x баллов к оценке категории t (по умолванию t="default"). X - произвольное число, не обязательно целое, не обязательно положительное. Также X может быть массивом

	чисел, тогда оценка будет выведена детализировано.
add_log_entry(t)	Добавить текст t в лог. Максимальный размер лога 100 тысяч символов. Если в результате дописывания лога величина будет превышена, более ранние строки будут потеряны.
add_achievement(t)	Добавить достижение (извещение) на страницу оценки. Если извещение с таким текстом уже было выведено, повторно оно выведено не будет. Максимальное количество извещений, которое может быть выведено - 1000.

Пример применения:

```
use 'strict';

// Вспомогательная функция для определения, находится ли аппарат в зоне интереса
function is_in_target_area(i) {
    let location = spacecrafts[i].motion.location;
    // далее так или иначе проверяем, что аппарат в зоне интереса
    // ...
}

// В начале симуляции проверяем, что аппаратов не более 4 , и выведем извещение, если это не так
function setup() {
    if (world.spacecrafts.length > 4) {
        runtime.add_achievement("Аппаратов в симуляции больше 4");
    }
}

function loop() {
    for (var i = 0; i < world.spacecrafts.length; i++) {
        let spcc = world.spacecrafts[i];
        if (!spcc.active) {
            continue;
        }
       if is_in_target_area(i) {
            runtime.add_score("default", 1/86400); // Если аппарат в зоне интереса, добавить 1/86400 балла к оценке. Если аппарат все время в нужной зоне это
```

```
даст ровно 10 баллов за сутки симуляции.
    }
}
```