Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Иркутский национальный исследовательский
технический университет»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине

«ИММЕРСИВНЫЕ ТЕХНОЛОГИИ»

Направление: 09.04.02 Информационные системы и технологии

Программа: Информационные технологии в промышленности

Методические указания для лабораторных работ составил

Директор института информационных технологий
и анализа данных, к.т.н А.С. Говорков

Иркутск 2024

Содержание

Предисловие и информация, необходимая перед началом работы с программой

Оборудование и системные требования для платформы Varwin — Документация Varwin 0.7.0 Beta

Как скачать пакеты Varwin Education и как с ними работать?

Оборудование и ПО:

Проектор/ интерактивная доска;

На каждого обучающего:

1. Персональный компьютер (с текстовым редактором, программой для презентаций, лицензия XRMS Varwin Education);
2. VR-HMD гарнитура (желательно каждому, но можно один шлем на двоих).

Также обратите внимание на то, что “Виртуальная реальность” - это быстро развивающаяся технология, поэтому важно постоянно актуализировать информацию и искать новые источники для использования их при подготовке.

Практическое занятие

0 - 15 минут	Преподаватель демонстрирует экран персонального компьютера. Обучающиеся наблюдают и запоминают, что происходит на экране. Фронтальная работа.
15 - 40 минут	Параллельная индивидуальная работа за компьютерами под контролем преподавателя. Преподаватель демонстрирует экран персонального компьютера. Обучающиеся наблюдают и повторяют то, что происходит на экране. Если возникают проблемы в ходе изучения, преподаватель по ходу занятия помогает обучающимся разобраться с возникающими трудностями.
40 - 45 минут	Учебная дискуссия. Групповая работа.

Кейсовое задание

0 - 10 минут	Получение кейса. Формирование собственного технического задания по проекту “*имя проекта*”.

На этом этапе обязательно нужно сформировать и зафиксировать техническое задание на проект

Шаблон технического задания:

1. Тематика/направленность проекта;
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы или же свои объекты);
3. Основной функционал приложения;
4. Дополнительные возможности приложения;
5. План-схема сцен, которые будут реализованы;

6. Каким образом будет реализован, удобный для пользователя, UX/UI - дизайн;
7. Какую полезную функцию для общества несет проект, или будет нести при дальнейшей его доработке.

10 - 40 минут	Разработка проекта согласно техническому заданию.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

40 - 45 минут	Сохранение проектов. Тестирование и демонстрация проектов на VR-HMD устройствах.

1 Знакомство с VR-технологиями

Занятие 1.1 Введение в технологию Виртуальная реальность

Цель:

Познакомить обучающихся с технологией Виртуальная реальность, RMS - системами, их историей и назначением.

Задачи:

- Сформировать представление о виртуальной и дополненной реальности, RMS-системах;
- Познакомить обучающихся с историей развития виртуальной реальности;
- Сформировать понимание функционирования существующих RMS-систем и VR-HMD устройств;
- Получить первый опыт Виртуальной реальности;
- Научить подготавливать комнату для использования VR, настраивать VR-HMD устройства и настраивать комнату для их использования;
- Познакомить обучающихся с RMS Varwin.

Методические материалы для подготовки к занятию:

1. История развития виртуальной реальности
2. Виртуальная реальность — Википедия
3. ГИД ПО VR В 2019. Какой шлем выбрать для ваших задач?
4. Standalone VR vs. PC VR: Key Differences
5. Как погрузиться в VR. Большой гайд по шлемам виртуальной реальности
6. Почему так трудно быть в VR: морская болезнь, отсутствие осязания и проблемы с проприоцепцией
7. AR vs VR vs MR: различия технологий и сферы применения
8. Установка и запуск платформы Varwin — Документация Varwin 0.7.0 Beta
9. Работа с Vive Focus — Документация Varwin 0.7.0 Beta
10. Quest and Rift S Setup
11. Setting up VIVE for the first time
12. Использование VR контроллеров — Документация Varwin 0.7.0 Beta

Структура презентации:

1. История развития VR/AR

   История развития виртуальной реальности

   a. 1837 год — первый стереоскоп
   b. 1956 год — Sensorama
   c. 1961 год — Headsight
   d. 1968 год — «Sword of Damocles»
   e. 1980 год — Eye Tap
   f. 1984 год — RB2, контроллеры First VR
   g. 1985 год — «Virtual Environment Display System»
   h. 1992 год — CAVE (первая комната виртуальной реальности)
   i. 1999 год — VirtuSphere
   j. 2012 год — Oculus VR

k. 2013 год — Google Glass
l. 2015 год — Microsoft Hololens

2. VR - что это за технология? Где мы её встречали и где она используется сейчас? Где она используется помимо шлемов виртуальной реальности?

   Виртуальная реальность — Википедия

   a. **Виртуальная реальность** (**ВР**, англ. *virtual reality*, *VR*, *искусственная реальность*) — созданный техническими средствами мир, передаваемый человеку через его ощущения: зрение, слух, осязание и другие. Виртуальная реальность имитирует как воздействие, так и реакции на воздействие. Для создания убедительного комплекса ощущений реальности компьютерный синтез свойств и реакций виртуальной реальности производится в реальном времени.
   b. Какие основные органы чувств задействуются при использовании VR?
   c. Применение виртуальной реальности (компьютерные игры, обучение, промышленность, видео)

3. Какие сейчас есть устройства на рынке? Какие есть сложности/проблемы в использовании VR?

   a. **Чем отличаются 3DOF 6DOF VR устройства?**
      Определения. 3DOF и 6DOF.
      DOF (degrees of freedom), степени свободы — это совокупность независимых координат перемещения и/или вращения.
      3DOF — пользователь может вращать головой в трех плоскостях и таким образом осматриваться вокруг.
      6DOF — дополнительно присутствует возможность смещаться по осям XYZ в пространстве, иными словами передвигаться благодаря смещению в реальном пространстве.
      При шести степенях свободы у пользователя появляется возможность совершать больше действий, например:
      Присесть;
      Наклониться, что-то рассмотреть;
      Обойти объект.
      Помимо прочего, предположительно адаптация в виртуальной реальности с 6DOF пройдет быстрее и легче из-за более естественного поведения камеры внутри симуляции.
   b. **Существующие актуальные VR-HMD решения:**
      Литература для подготовки к занятию:
      ГИД ПО VR В 2019. Какой шлем выбрать для ваших задач?
      Рассказать какие персональные компьютеры нужны для полноценного VR. Или как использовать мобильный VR. Преимущества и недостатки.
      PC VR, STANDALONE VR - отличия
      Литература для подготовки к занятию:
      Standalone VR vs. PC VR: Key Differences
      Отличие трекинга на базовых станциях и оптического трекинга

(Vive vs Rift S, пункт 2.2.)

Литература для подготовки к занятию:

Как погрузиться в VR. Большой гайд по шлемам виртуальной реальности

Рекомендация: На примере Вашей гарнитуры покажите как работает система отслеживания шлема и контроллеров. Как настраиваются основные функции шлема.

Актуальные решения на данный момент:

- Oculus Rift (CV1 / S)
- HTC (Vive / Vive pro)
- Valve Index
- WMR шлемы
- HTC VIVE Focus/Focus Plus
- Oculus Quest

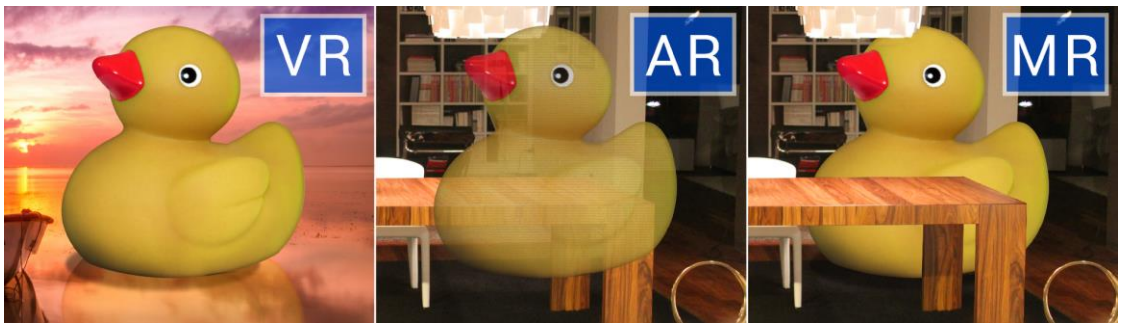
c. **Возможные проблемы при использовании VR:**

Литература для подготовки к занятию:

Почему так трудно быть в VR: морская болезнь, отсутствие осязания и проблемы с проприоцепцией

- Морская болезнь
- Сложность использования в очках для коррекции зрения
- Головокружение

4. Основные отличия VR/AR/MR. Чем отличается VR от всех других технологий. Делать упор на сравнение других технологий с VR.



Литература для подготовки к занятию:

AR vs VR vs MR: различия технологий и сферы применения

5. Немного об RMS-системах:

Для создания VR-приложений наибольшей популярностью пользуются игровые движки (или игровые среды разработки), такие как Unity3D и Unreal Engine. Однако, чтобы научиться работать с этими инструментами требуется многолетнее обучение, знание тонких особенностей этих игровых движков и владение языками программирования, такими как C# или C++. Но уже сейчас появляются программы снижающие порог входа в разработку VR-контента - RMS-системы (RMS - англ. Reality Management System или Системы управления [виртуальной] реальностью. RMS-система - это конструктор,

который позволяет разрабатывать проекты без знания программирования на основе шаблонов объектов и локаций, созданных ранее профессиональными разработчиками в игровых движках, таких как Unity3D. В частности, сегодня мы познакомимся с платформой Varwin - это хорошая точка старта для начинающего VR-разработчика, для базовых принципов создания VR-проектов и для дальнейшего перехода на более глубокие уровни разработки.

6. Поговорить о сферах в которых развивается VR. Какие есть профессии в VR индустрии. Кем в будущем смогут стать обучающиеся.

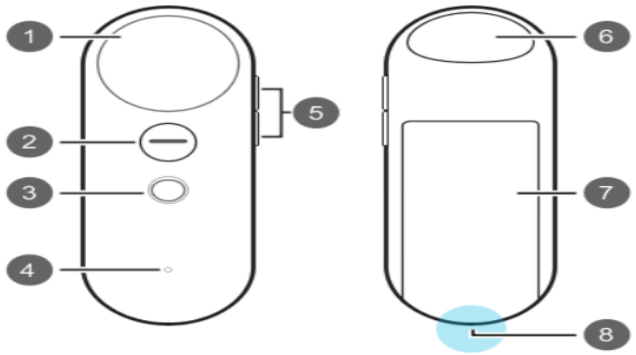
   Краткое введение в профессиональные перспективы:

   Многие работы в рамках компетенции могут выполняться специалистом в качестве индивидуального предпринимателя:

   - Работа в качестве freelance-разработчика по специальности Unity3D developer (средняя ставка по рынку 20-30$ за час).
   - Открытие собственной digital-студии по разработке VR-контента:
   - Игровые приложения для широкого потребительского рынка (B2C);
   - Индивидуальная разработка проектов на заказ для бизнеса (B2B): тренажеры, выставочные решения, образовательные приложения и др.
   - Разработка собственного VR-продукта: VR-социальная сеть, конструкторы и др.
   - Открытие образовательных курсов по направлению “Виртуальная реальность”

     1. Настройка VR-HMD устройства для использования (и, если необходимо, подключение к ПК и настройка на ПК)
        Ссылка на актуальную документацию Varwin:
        Установка и запуск платформы Varwin — Документация Varwin 0.7.0 Beta
     2. Настройка комнаты для работы с VR-HMD устройством
        Ссылка на актуальную документацию Varwin:
        Работа с Vive Focus — Документация Varwin 0.7.0 Beta
        Литература для подготовки к занятию:
        Quest and Rift S Setup
        Setting up VIVE for the first time

     1. Демонстрация возможностей Varwin XRMS в VR. Показать несколько самых интересных проектов. На этом этапе нужно максимально показать возможности XRMS и впечатлить обучающихся.
     2. Изучение управления в VR режиме Varwin XRMS.
        Если вы используете Vive Focus, то следуйте инструкции:

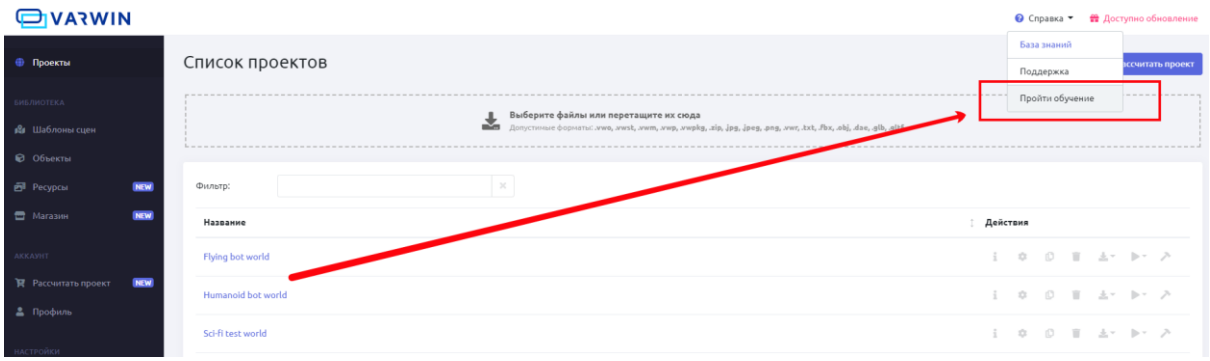


1	Сенсорная панель
2	Кнопка «Приложения»
3	Кнопка «Домой»
4	Индикатор состояния
5	Кнопки «Громкость»
6	Курок
7	Отсек для батареек
8	Отверстие для браслета

- Для **взаимодействия** с объектами нажмите **курок**.
- Для **телепортации** нажмите и удерживайте **сенсорную панель**, чтобы прицелиться и отпустите, чтобы телепортироваться.
- Для дискретного поворота вправо/влево нажимайте на правую/левую части сенсорной панели.
- Чтобы **взять** или **отпустить** объект, нажмите кнопку **«Приложения»**.
- Чтобы выйти из проекта на экран подключения, нажмите и удерживайте в течении двух секунд кнопку **«Приложения»**.
- Чтобы вызвать системное меню Vive Focus, нажмите кнопку **«Домой»**.
- Для взаимодействия с элементами интерфейса нажмите **сенсорную панель** или **курок**.

Если вы пользуетесь ПК-гарнитурой, то пройдите обучение.

Для этого нажмите на кнопку “**Справка**” и выберите пункт выпадающего меню “**Пройти обучение**”.



***Примечание**: если у вас нет VR-гарнитуры, не расстраивайтесь, очень скоро она будет у всех, как когда-то стал неотъемлемой вещью мобильный телефон. А пока пройдите созданный проект в режиме просмотра на ПК.*

Ссылка на актуальную документацию Varwin:

Использование VR контроллеров — Документация Varwin 0.7.0 Beta

Примечание: Если НЕ все обучающиеся успеют попробовать технологию виртуальной реальности, то далее на своих рабочих местах они должны будут самостоятельно попробовать готовые VR-проекты. Если есть дополнительное время чтобы попробовали все, то лучше выделить его на этом занятии.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое виртуальная реальность?
2. Что такое Varwin XRMS?
3. Назовите основные VR-HMD устройства, которые используются в наше время.
4. Назовите основные моменты управления Varwin XRMS в режиме VR.
5. Что такое DOF? Перечислите все 6 DOF.
6. Что такое трекинг VR-устройства? И какие виды трекинга бывают?

Занятие 1.2 Desktop-редактор Varwin

Цель:

Познакомить обучающихся с Desktop-редактором RMS Varwin.

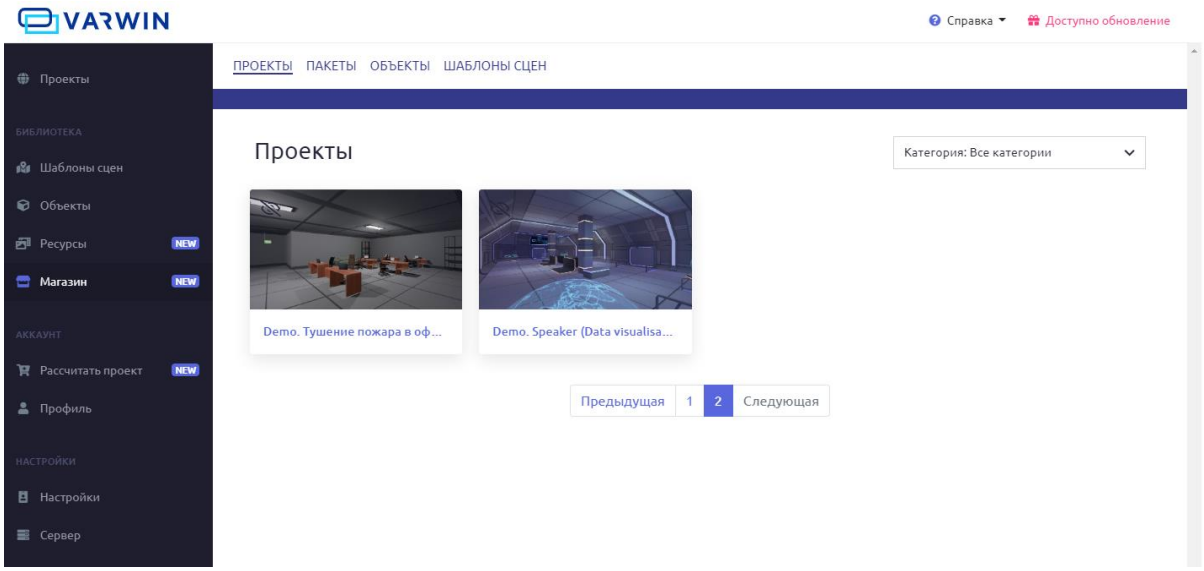
Задачи:

- Изучить интерфейс RMS Varwin;
- Научить обучающихся создавать проекты, работать с шаблонами сцен и использовать несколько сцен в проекте;
- Изучить интерфейс Desktop-редактора RMS Varwin;
- Сформировать понимание параметров позиционирования объектов;
- Научить обучающихся сохранять и загружать проекты.

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #1

1. **Изучение интерфейса Varwin XRMS**



Для начала откроем XRMS Varwin и посмотрим какие основные вкладки присутствуют в интерфейсе программы. Запомним с чем мы будем взаимодействовать.

Проекты - в этой вкладке Вы будете видеть проекты которые разработали сами или загрузили в XRMS Varwin.

Шаблоны сцен - здесь будут отображены шаблоны сцен, готовые для использования в ваших проектах.

Объекты - в этой вкладке отображены основные объекты из пакетов, которые Вы сможете скачать во вкладке “Магазин”

Ресурсы - здесь Вы будете видеть мультимедиа файлы, распространенных форматов, которые можно загружать в платформу напрямую для дальнейшего использования в проектах.

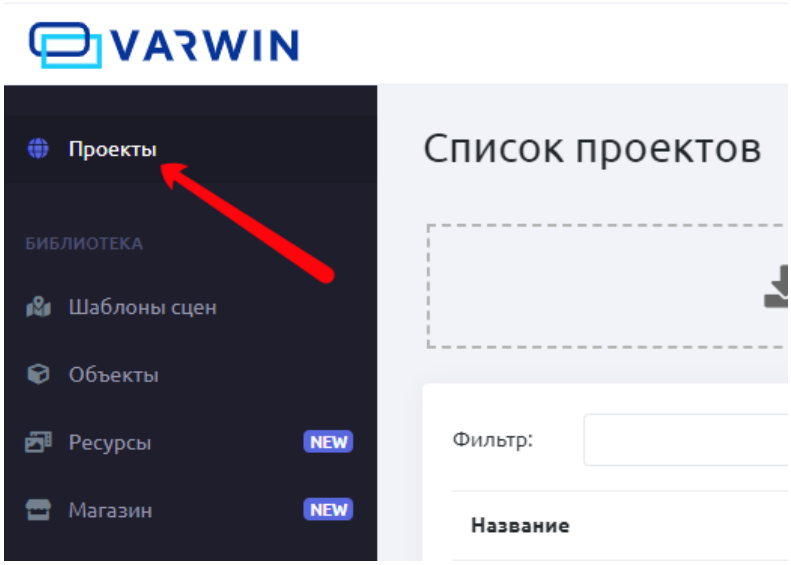
Магазин - в магазине Вы можете скачать дополнительные пакеты объектов, которые будут доступны для использования в Ваших проектах.

2. Создание проекта и выбор шаблона сцены

Ссылка на актуальную документацию Varwin:

Редактирование проекта — Документация Varwin 0.7.0 Beta

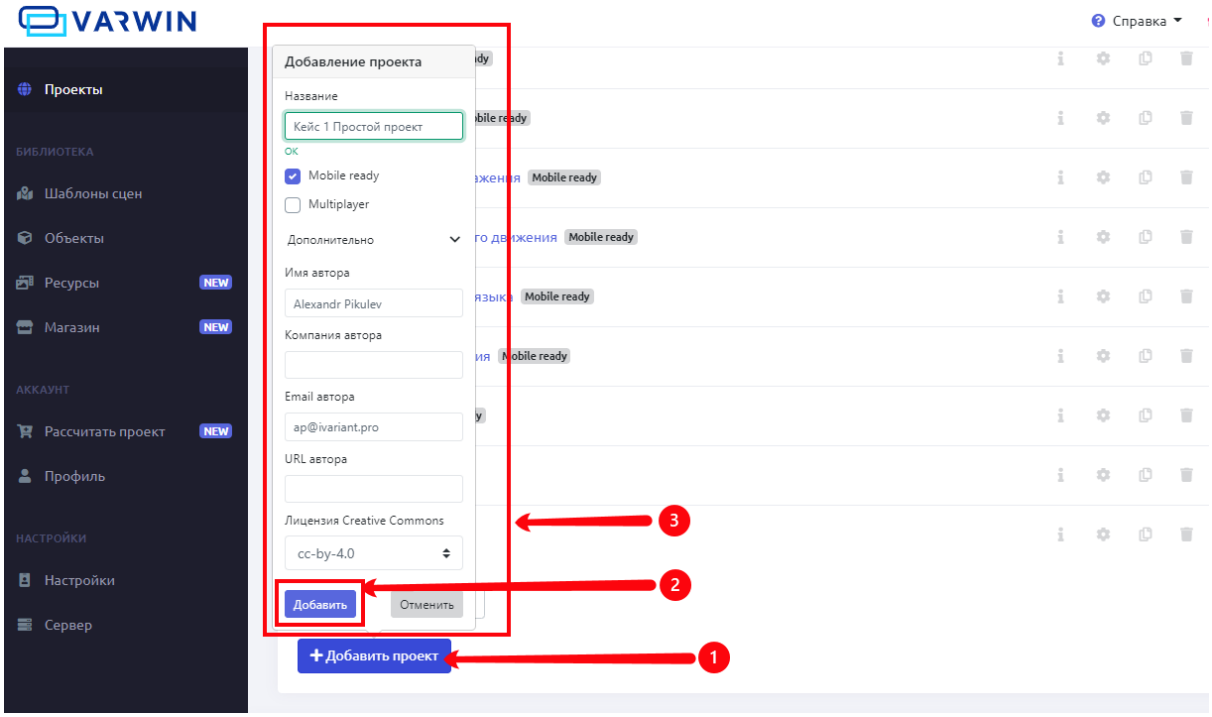
Теперь научимся создавать новый проект: для этого перейдите во вкладку проекты, которая находится в главном навигационном меню.



Нажмите на кнопку “Добавить проект” (1).
Введите название проекта и заполните информацию о нем по желанию.

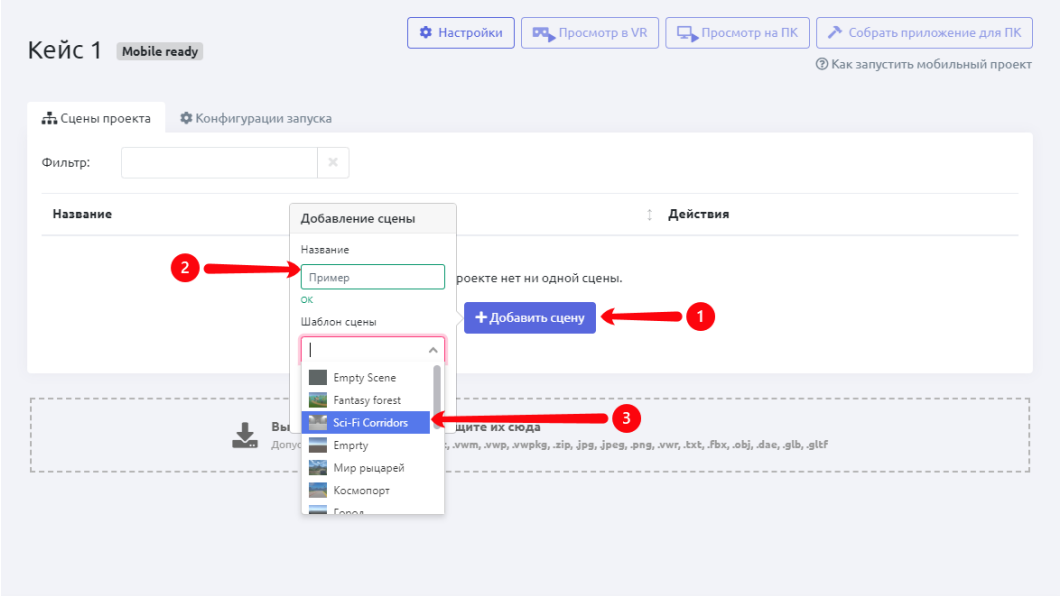
Вы можете установить флажок “Mobile ready”, если желаете, чтобы ваш проект запускался на мобильных шлемах (Vive Focus, Oculus Quest), а также на смартфонах с операционной системой Android.

После заполнения полей нажмите на кнопку “Добавить” для создания нового проекта (2).



Добавление сцены

Вы автоматически попадаете в окно со сценами проекта, но поскольку этот проект новый, ни одной сцены в нем еще не присутствует.
Для добавления новой сцены нажмите на кнопку “Добавить сцену” (1).
Введите название сцены (2) и выберите доступный шаблон сцены из вашей библиотеки (3).

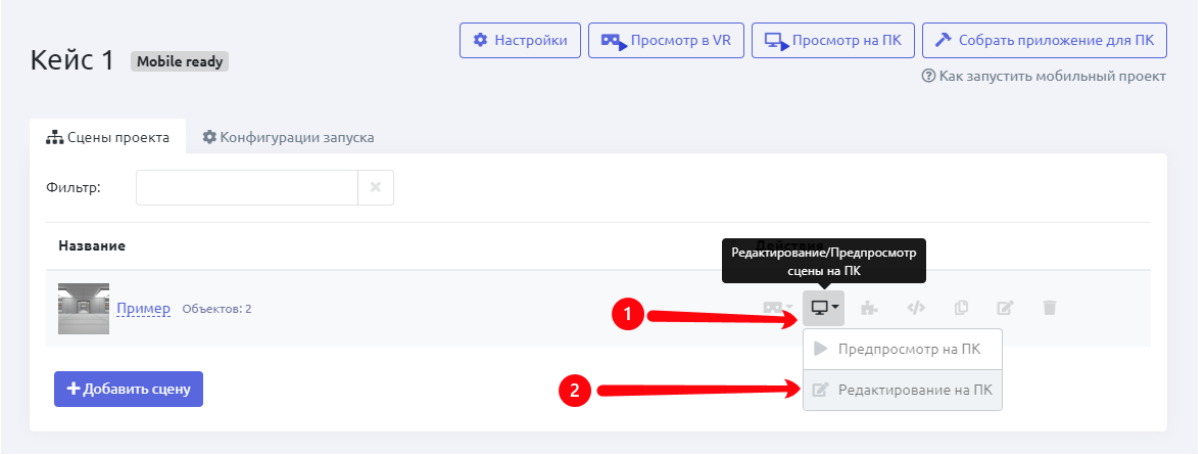


3. Изучение интерфейса Desktop - редактора

Ссылка на актуальную документацию Varwin:
Редактирование сцены — Документация Varwin 0.7.0 Beta

Запуск Desktop-редактора

После добавления сцены нажмите на иконку “Редактирование/Предпросмотр сцены на ПК” (1) и выберите “Редактирование на ПК” (2).



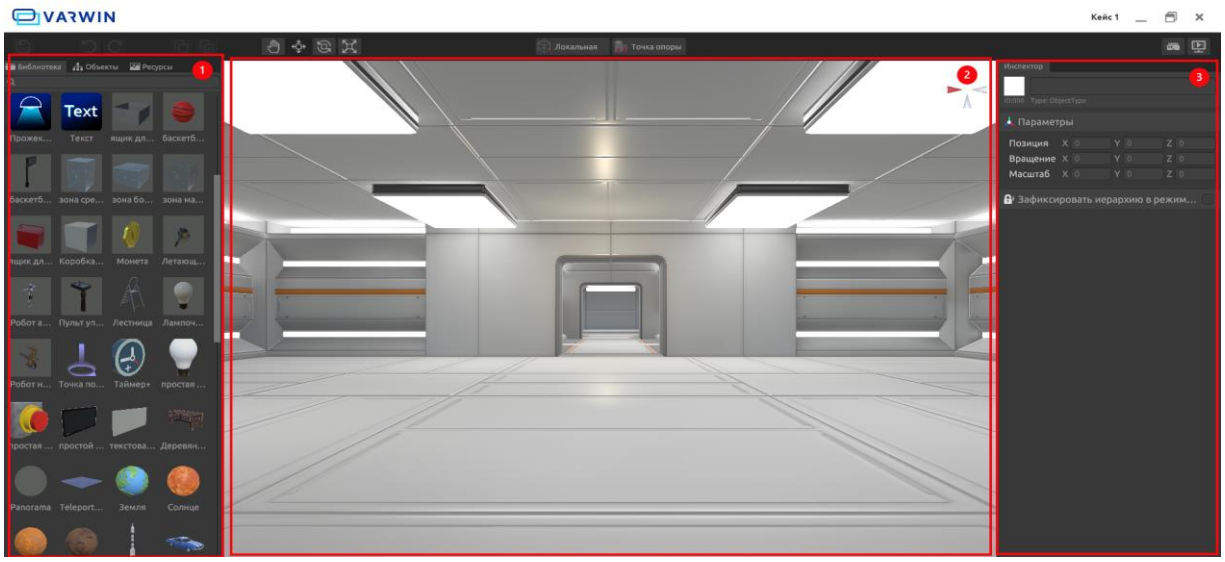
Базовый интерфейс Desktop-редактора.

Вы открыли редактор.

Основные окна редактора - это ***библиотека*** (1), в которой можно увидеть доступные для размещения на сцене объекты.

Окно навигации по сцене (2). Для смены ракурса камеры зажмите правую кнопку мыши и вращайте, для смены позиции камеры используйте клавиши “WASD”, для перемещения с ускорением зажмите клавишу “Shift”.

Совет: *Освойтесь с управлением внутри редактора, пролетев всю сцену.*



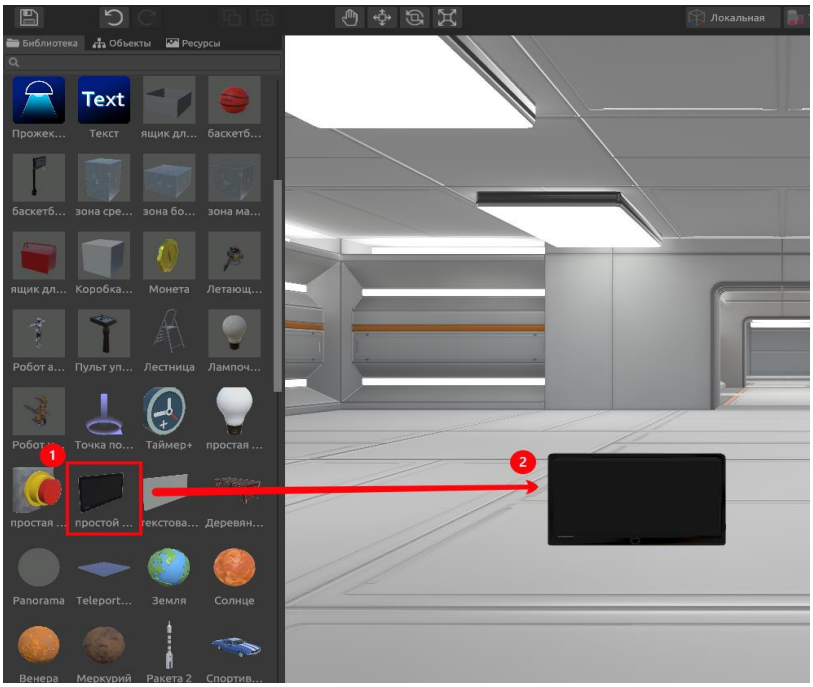
Третье окно - это ***инспектор*** (3), в нем содержатся параметры и свойства выделенного объекта, оно понадобится на чуть позже.

1. **Создание проекта, выбор шаблона сцены.**
2. **Знакомство с объектами (простой дисплей, простая лампочка** и **простая кнопка**)

Совет: Особо рвущимся вперед обучающимся можно дать задание на усвоение задач занятия. Задание на расстановку объектов из пакета Education по определенному шаблону сцены и сохранение своего проекта.

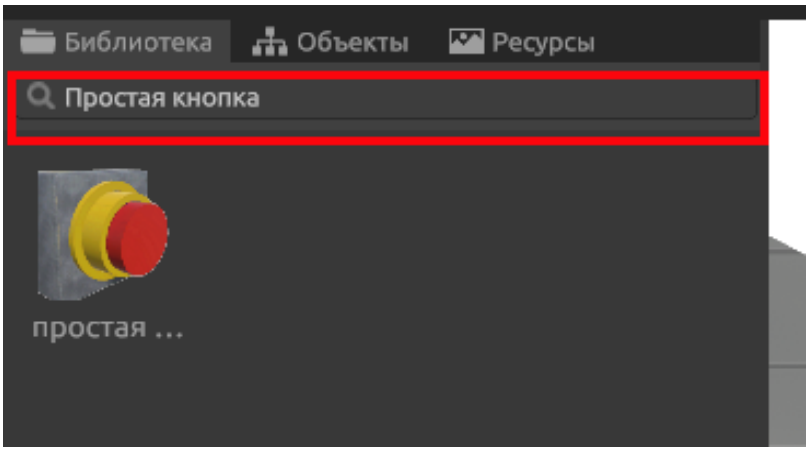
Размещение объектов из библиотеки.

Для размещения объекта на сцене выберите объект в библиотеке, наведите на него курсор мыши (в нашем примере это объект *“**простой дисплей**”* (1)), зажмите левую кнопку мыши и перетащите его в необходимую область окна навигации по сцене (2) (методом “***drag and drop***”, как мы перемещаем файл из одной папки в другую).



Таким же образом разместите на сцене, кроме **простого дисплея**, также **простую лампочку** и **простую кнопку.**

***Совет**: для быстрого поиска объектов по названию воспользуйтесь строкой поиска! Просто введите название объекта в строку и немного подождите.*



3. Изучение работы перемещения/вращения/масштабирования объектов

Параметры позиционирования объектов.

Теперь давайте научимся перемещать объекты по сцене, вращать их и масштабировать.

Для этого щелкните левой кнопкой мыши на один из объектов, размещенных на сцене.

Как вы можете наблюдать, появляются оси координат (1).

И в инспекторе появляются параметры этого объекта, относительно осей “XYZ” (2):

1. Позиция;
2. Вращение;
3. Масштаб.

Позиция

Мы можем изменить позицию объекта просто потянув мышкой за одну из координатных осей. (горячее сочетание клавиш “CTRL + W”)

Также мы можем ввести координаты объекта вручную в инспекторе в строке “***позиция***”. Это может потребоваться для более точного выравнивания объектов.

Расположите объекты так, как вы считаете удобным и эстетичным, например у стены.

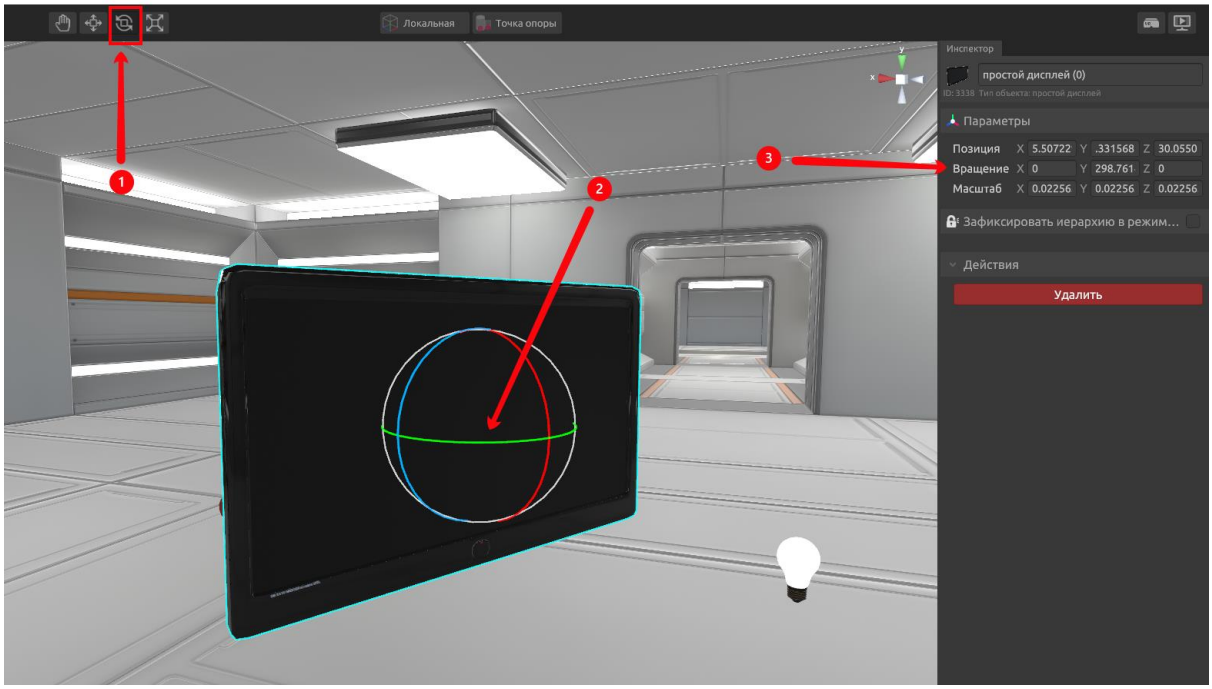
Вращение

Для вращения объекта нажмите нажмите на кнопку “Вращение объекта” (1) или горячее сочетание клавиш “CTRL + E”.

Ось координат сменится на ось вращения (2).

Для поворота объекта потяните мышкой за одну из этих осей.

Параметры вращения можно также ввести вручную в инспекторе (3).



Попробуйте повернуть один из объектов разными способами, чтобы закрепить новые знания.

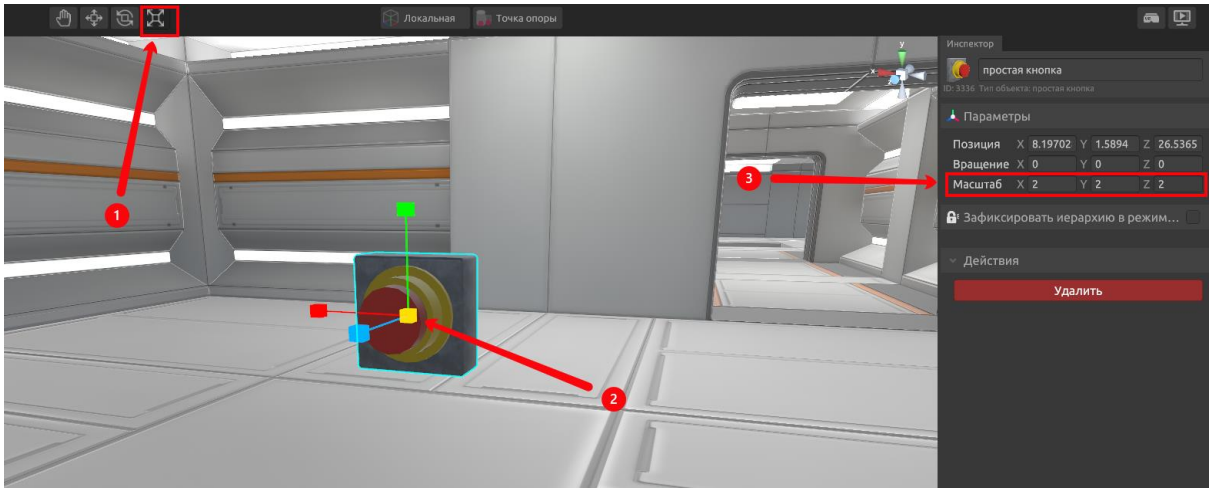
Масштабирование

Для масштабирования объекта нажмите нажмите на кнопку “Масштабирование объекта” (1) или горячее сочетание клавиш “CTRL + R”. Появится ось масштабирования (2).

Чтобы увеличить объект относительно одной из осей потяните мышкой за соответствующую ось.

Чтобы масштабировать объект пропорционально, относительно каждой из осей, наведите курсор на центральный кубик и потяните мышкой в любую сторону.

Параметры масштабирования можно также ввести вручную в инспекторе (3).



Давайте увеличим кнопку в два раза, чтобы на нее было удобно нажимать. Для этого просто введем цифру 2 в параметрах масштабирования для каждой из осей “XYZ”.

Чтобы вернуться к изменению позиции нажмите сочетание “CTRL + W” или выберите соответствующую иконку на верхней панели (1).

***Совет**: еще один способ перемещения камеры можно активировать, если нажать иконку “**руки**” (2). В этом случае вы сможете перемещать камеру в горизонтальной и вертикальной плоскости зажав левую кнопку мыши.*

4. Изучение способов сохранения и загрузки сцены. Сохранение проектов, проверка успешного сохранения. (Важно чтобы дети не теряли свои проекты)

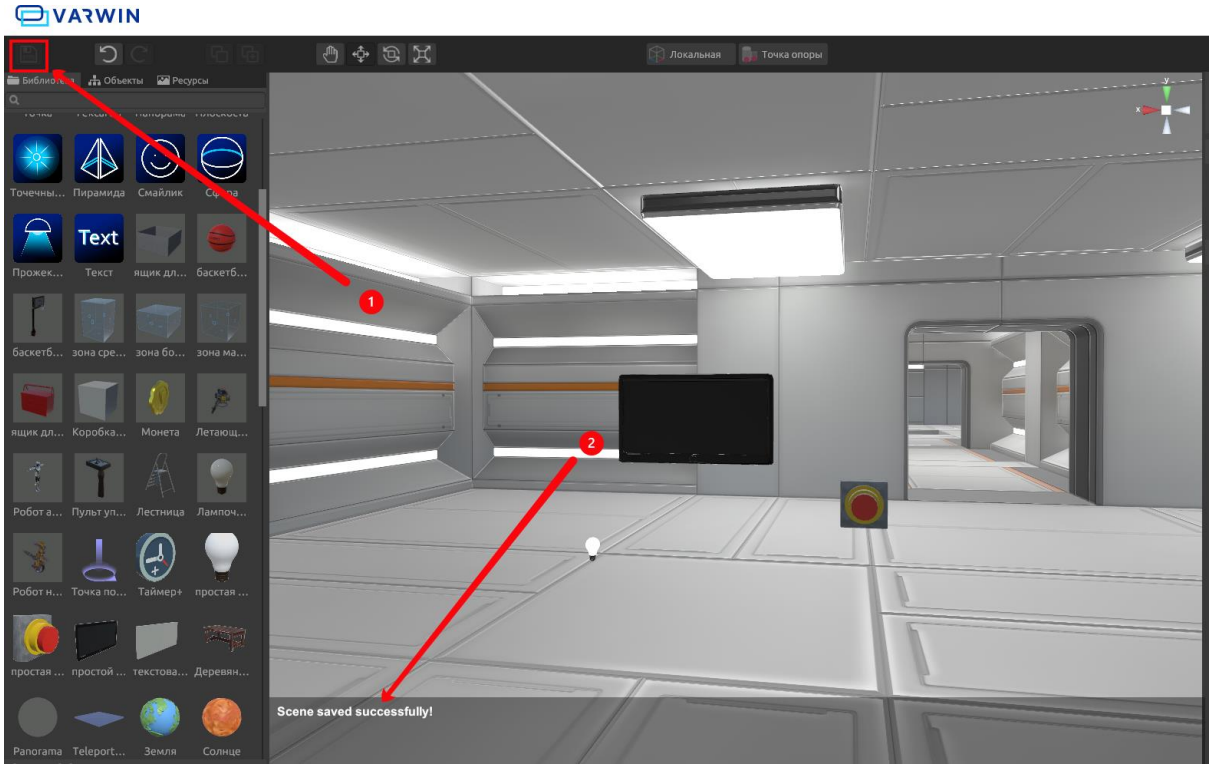
Сохранение сцены с объектами, завершение работы с редактором сцен

После того как мы закончили размещение объектов на сцене и настройку их параметров можно переходить к созданию логики взаимодействия.

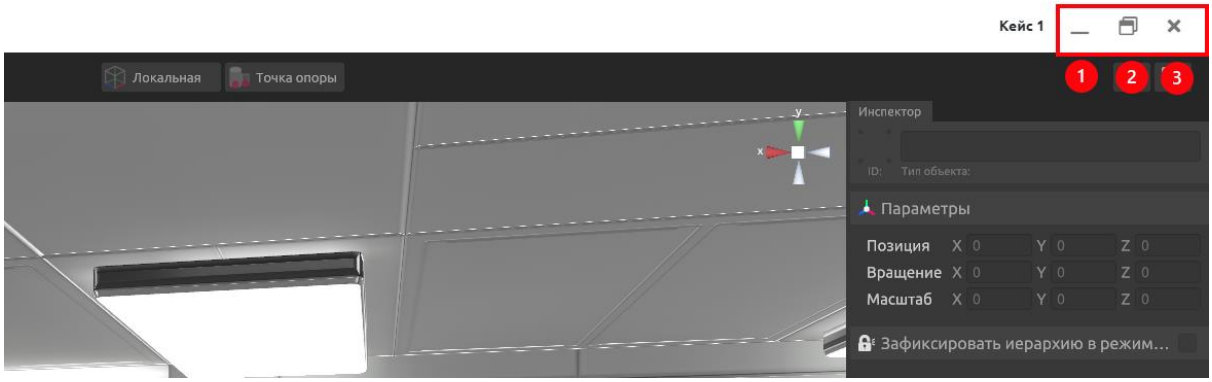
Но перед этим **важно** сохранить созданную сцену.

Для этого нажмите на иконку “***дискеты***” (1) на верхней панели или горячее сочетание клавиш “CTRL + S”.

После этого должно появиться сообщение об успешном сохранении сцены (2). При этом иконка “***дискеты***” станет серой (неактивной) до внесения следующих изменений в сцену.



Далее проект можно закрыть (3), свернуть в окно (2) или свернуть полностью (1).



***Совет**: для быстрого доступа к Desktop-редактору сворачивайте проект, а не закрывайте. В этом случае будет удобно и быстро переключаться между редактором логики и редактором сцен без необходимости ожидания загрузки. Проект, свернутый в окне, позволит увидеть объекты для которых вы формируете логику.*

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. В каком разделе XRMS Varwin можно скачать основные пакеты объектов?
2. Назовите 3 инструмента позиционирования объектов в пространстве.
3. Назовите 3 основных окна Desktop редактора XRMS Varwin.

Занятие 1.3 Редактор логики Varwin

Цель: Познакомить обучающихся с редактором логики “Blockly” RMS Varwin.
Задачи:

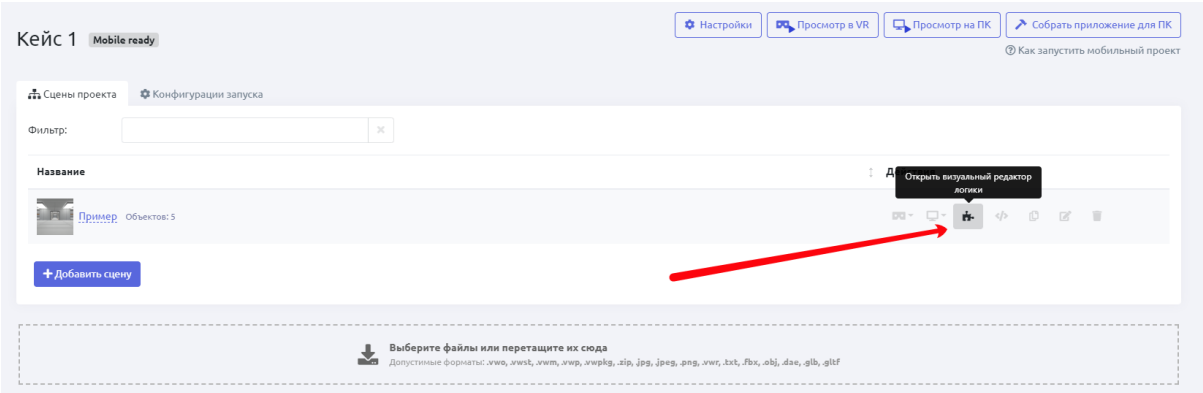
- Изучить основной интерфейс редактора логики “Blockly”;
- Сформировать понимание о существующих типах логических блоков;
- Усвоить принципы соединения боков и создания логики взаимодействия между объектами;
- Изучить управление в Desktop-режиме;
- Научить тестировать проекты, проверять работоспособность логики в проекте, исправлять ошибки логики взаимодействия и взаимодействовать с консолью вывода ошибок/предупреждений.

Методические материалы для подготовки к занятию: Пошаговая инструкция по разработке кейсов #1.

1. Изучение интерфейса Blockly

Запуск редактора логики

Для запуска редактора логики для определенной сцены нажмите на кнопку “Открыть визуальный редактор логики”.



Базовый интерфейс редактора логики

***Примечание**: В данном проекте мы не будем разбирать детально все особенности интерфейса редактора логики, типы логических блоков, вкладки и прочее, только базовые функции. Остальные детали будут разбираться в следующих модулях.*

(1) Это панель выбора типов блоков. В данном кейсе нам понадобятся только типы “Логика” и “Объекты”.
(2) Из редактора логики можно запустить редактор или проект (па ПК или в VR).
(3) Это инструменты для центрирования и масштабирования, а также корзина для удаления блоков.

Основное пространство - это **рабочая область редактора логики.**

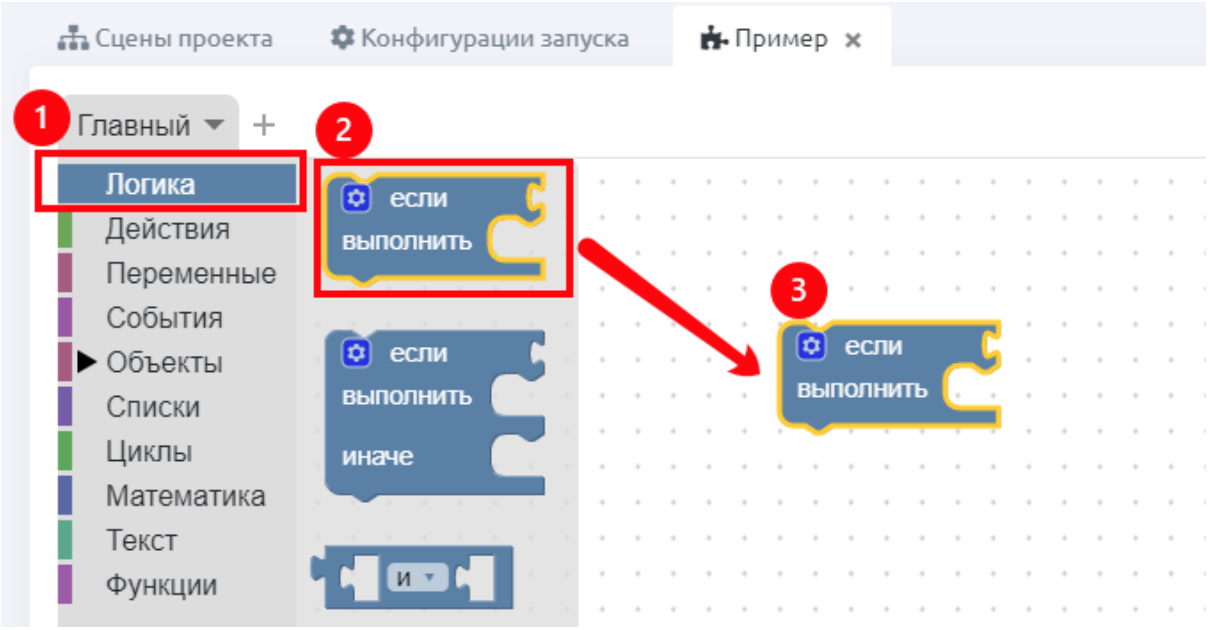
2. Демонстрация готового проекта “Лампочка”

Создание простой логики

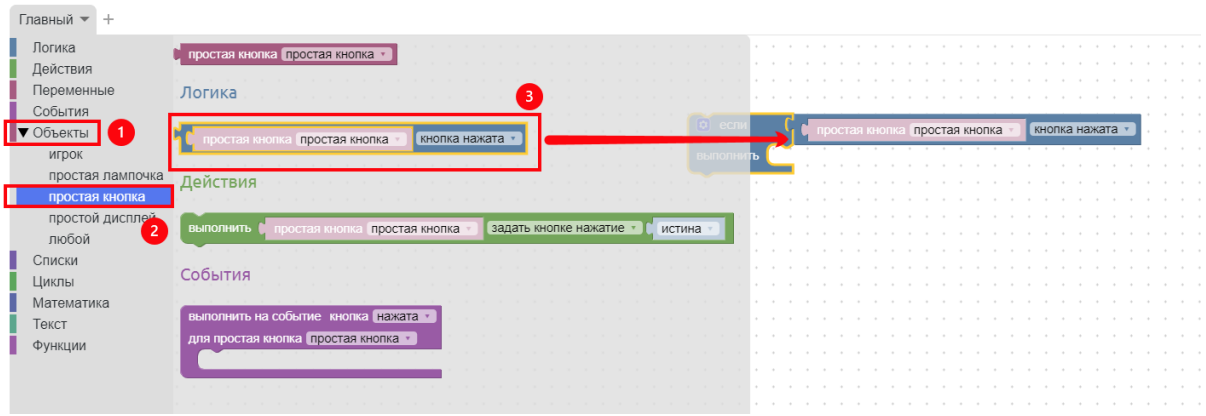
Не погружаясь в детали, давайте создадим следующую логику взаимодействия для наших объектов на сцене:

Пока кнопка не нажата - лампочка не горит, а на дисплее висит надпись “Нажми на кнопку!”, если мы нажимаем на кнопку, то лампочка загорается и на дисплее появляется текст “Привет, мир!”.

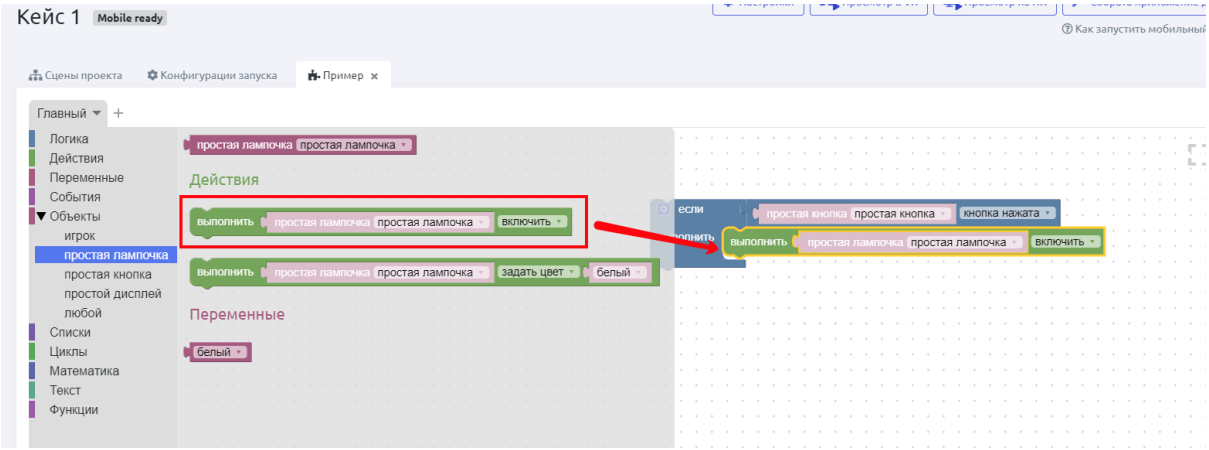
Для этого выберем логический тип “Логика” с помощью левой клавиши мышки (1).
Далее наводим курсор на блок “Если - Выполнить” (2).
Зажимаем левую клавиши мыши и перетаскиваем его в рабочую область редактора логики (3).



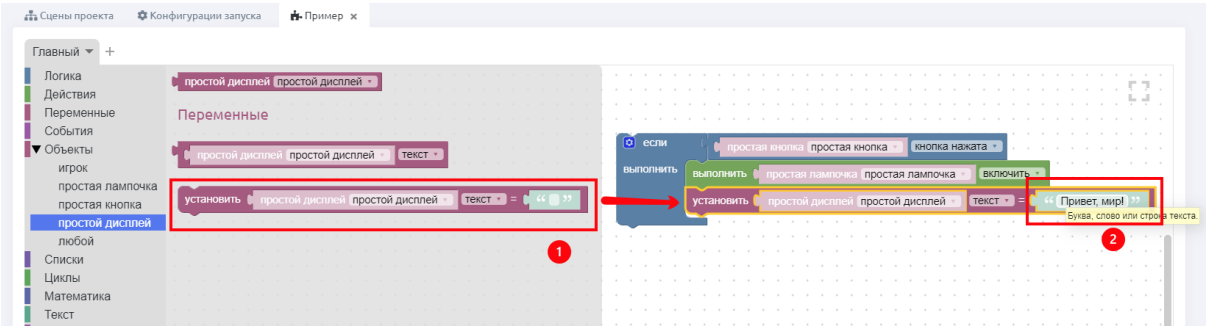
Далее выбираем тип “Объекты” (1).
Как вы видите, здесь те самые объекты, которые мы ранее добавили на сцену.
Выберем простую кнопку (2) и блок “кнопка нажата” (3).
Перетянем блок в рабочую область и соединим с блоком “Если - Выполнить” как-будто соединяем части пазла.



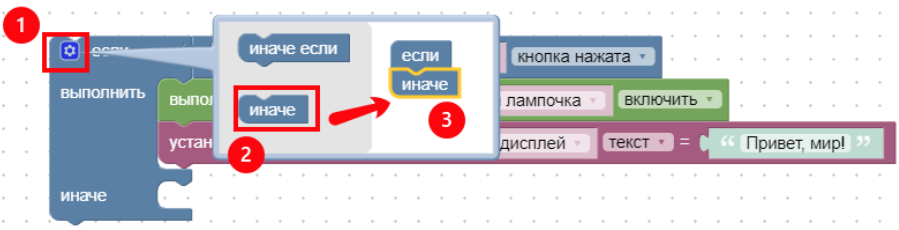
Такую же операцию делаем для блока “включить лампочку” и для блока дисплея “установить текст”.



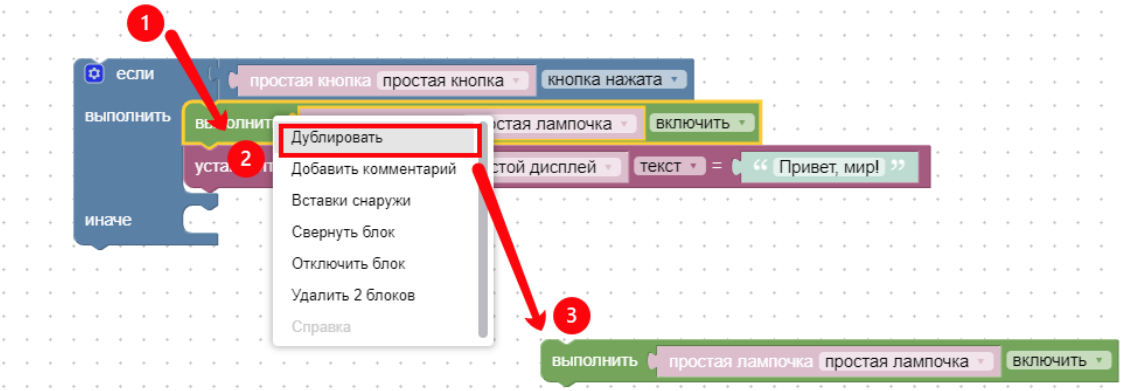
В блок “установить текст” в рабочей области редактора логики вписываем текст “Привет, мир!”.



Далее вам необходимо задать события в случае, если кнопка будет не нажата.
Для этого нажимаем на иконку “шестеренка” (1).
Выбираем блок “Иначе”.
Перетаскиваем его под блок если в подменю “шестеренки” (3).

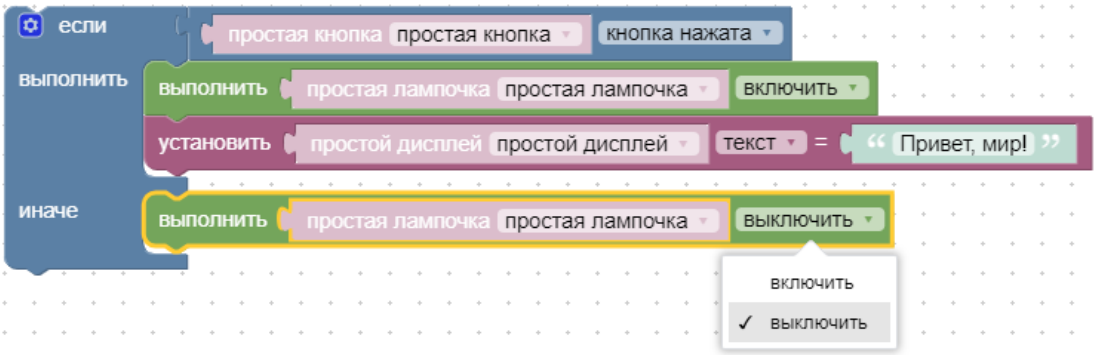


Следующим шагом научимся дублировать блок.
Щелкните правой кнопкой мыши на блок (1).
Появится выпадающее меню, выберете в нем пункт “Дублировать” (2).
Перетащите необходимый блок в нужное положение (3).

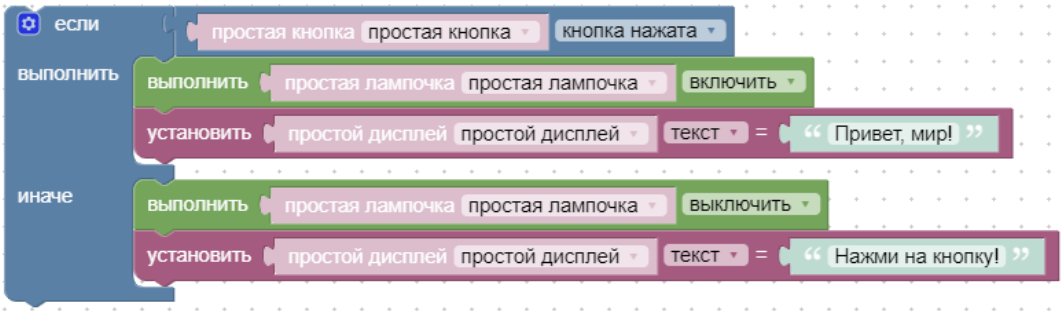


***Совет**: блоки можно копировать быстрее, если использовать горячие клавиши "CTRL + C" -> "CTRL + V".*

Теперь вставим блок с лампочкой напротив "**иначе**" и выберем состояние "**выключить**".
Это значит, что если кнопка будет **НЕ** нажата, то лампочка будет находиться в выключенном состоянии.



Финальная логическая конструкция этого проекта будет выглядеть как на картинке ниже:



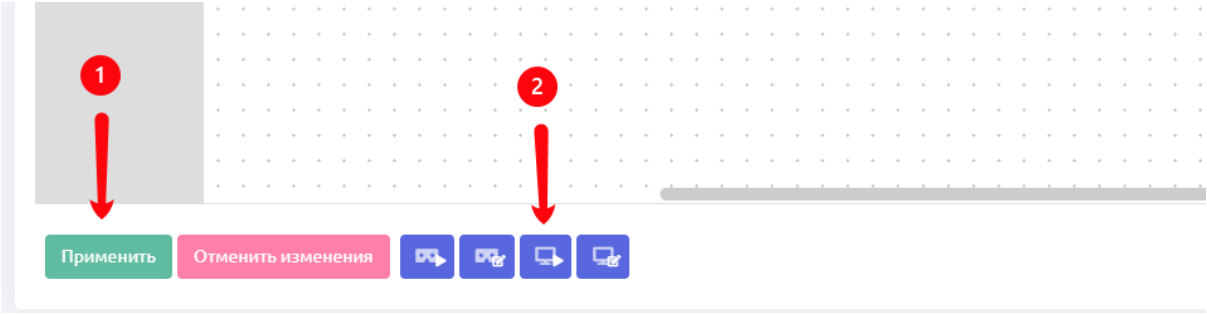
***Совет**: если вы сделали что-то неправильно, то можно удалить лишний блок в выпадающем меню, либо нажать сочетание "CTRL + Z" - это отменит последнее совершенное действие.*

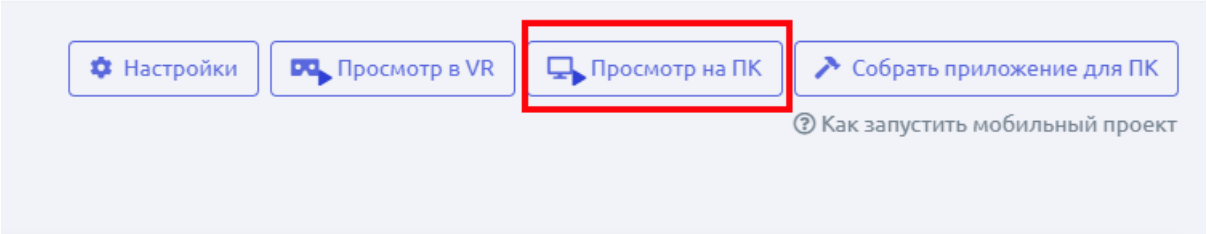
Сохранение логики и запуск проекта.

Логика создана, перед запуском проекта обязательно необходимо ее сохранить.

Для этого нажмите на кнопку "Применить" (1). После применения логики кнопка станет неактивной до внесения следующих изменений.

Данный проект мы будем запускать в Desktop-режиме. Для этого можно нажать на кнопку (2) в редакторе логики, либо кнопку "Просмотр на ПК" в верхнем меню проекта.



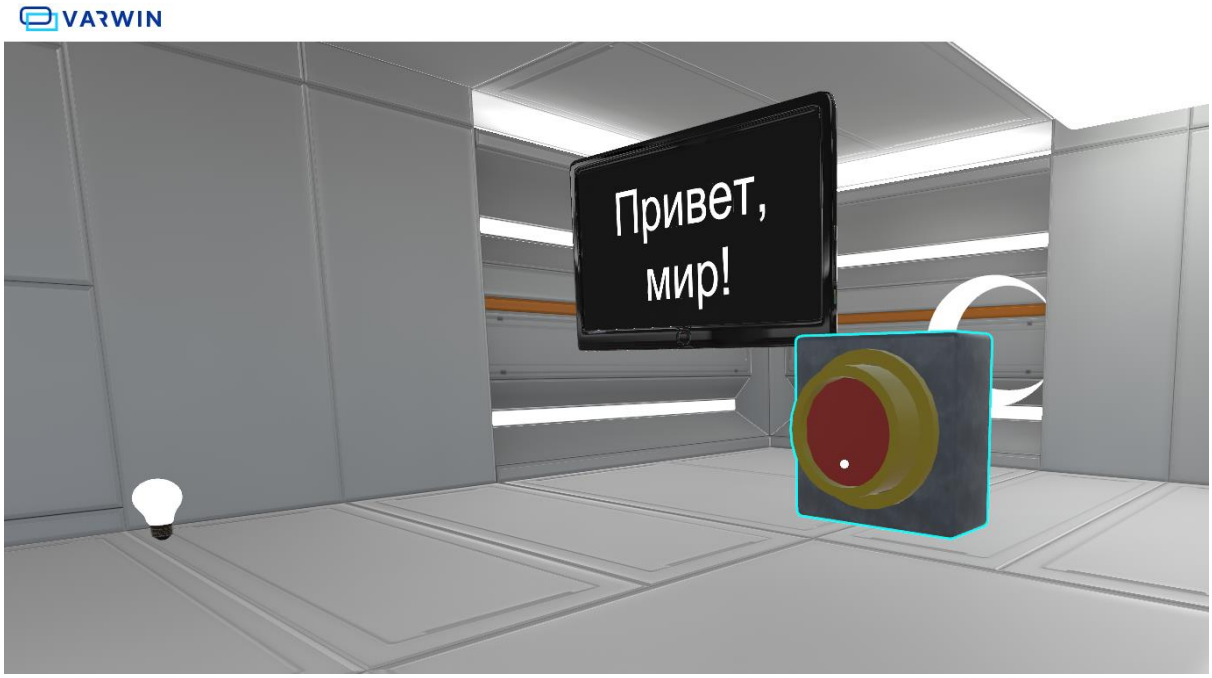


3. **Управление в desktop режиме**

A. "стрелки" или клавиши "w, a, s, d" – перемещение.
B. "мышь" – обзор.
C. "CTRL" – присесть.
D. «левая кнопка мыши» – взаимодействие с объектами (например, нажать на кнопку).
E. «правая кнопка мыши» – взять предмет в руки. Повторное нажатие на правую кнопку мыши отпускает предмет. В этом сценарии эта кнопка нам не понадобится, но запомните на будущее!
F. "Q" - активировать телепорт.
G. "shift" - ускорение.

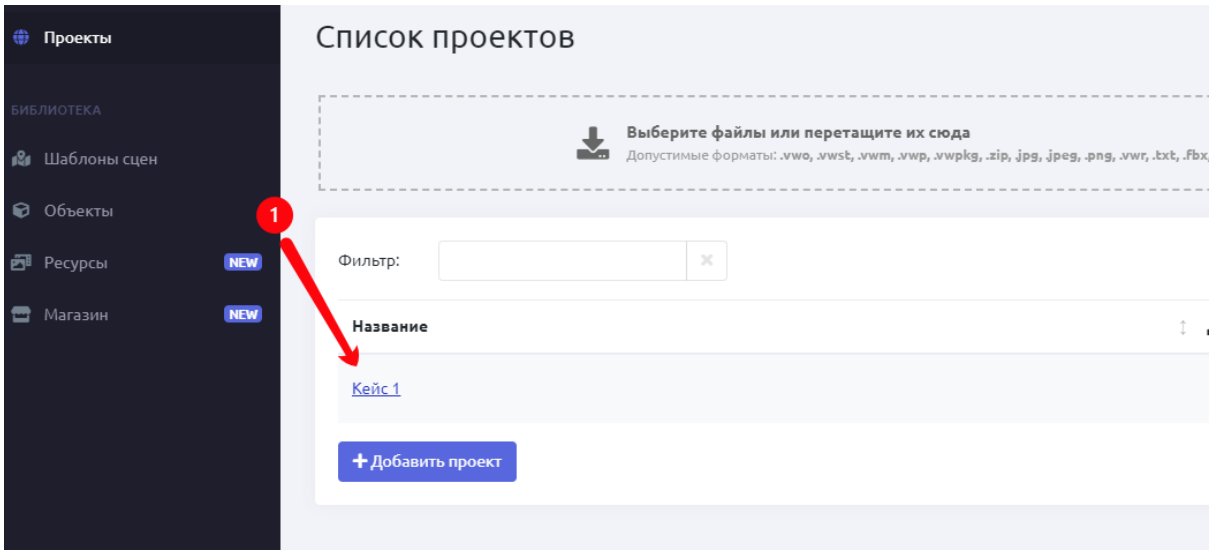
4. **Демонстрация готового проекта "Лампочка" и его логики в Blockly**

Если все работает правильно, то при нажатии на кнопку (левая кнопка мыши), вы увидите примерно такую картинку:



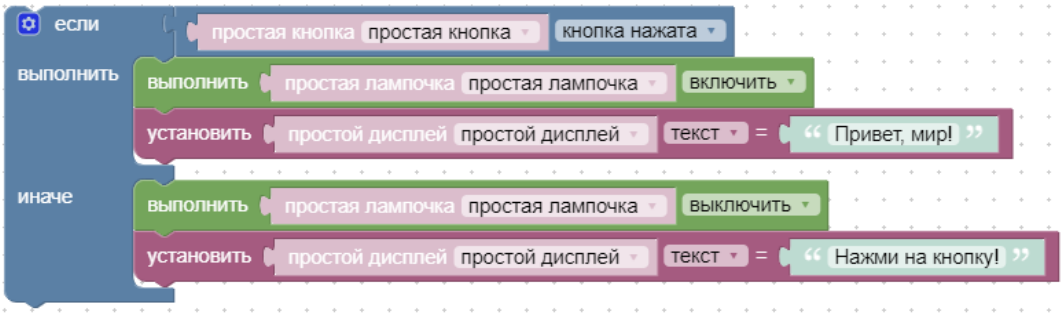
Обучающиеся самостоятельно пробуют собрать продемонстрированный проект. Преподавателю обязательно нужно зафиксировать финальный вариант логики проекта на проекторе/ интерактивной доске, чтобы у обучающихся в любой момент была возможность посмотреть правильность выполнения.

1. **Открытие ранее сохраненного проекта**

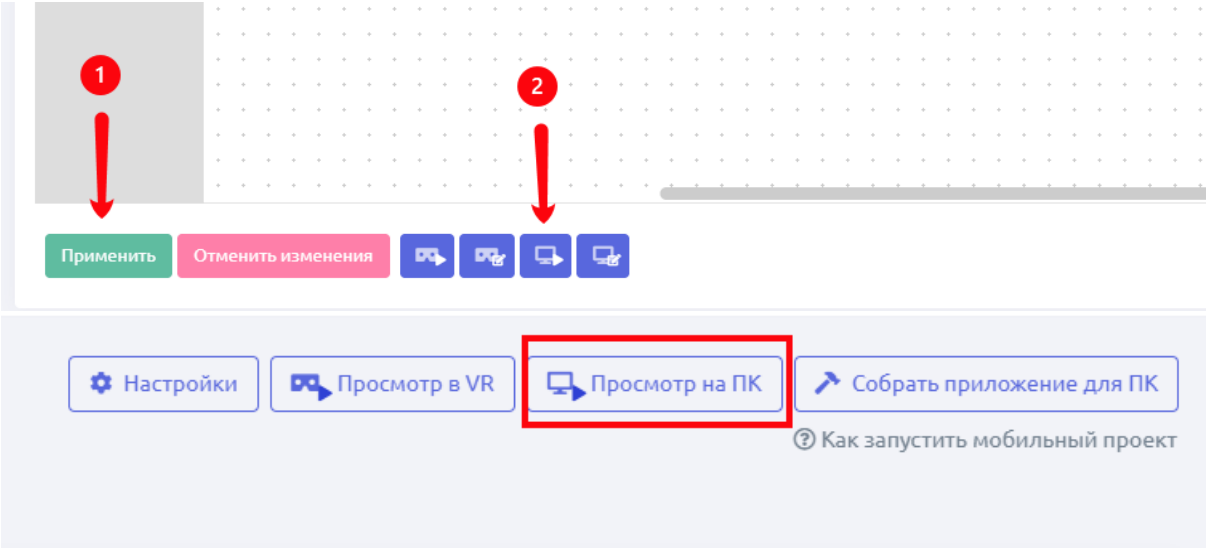


2. **Проверка размещения объектов** (**простой дисплей**, **простая лампочка** и **простая кнопка**) на сцене. Объекты должны быть аккуратно расставлены на локации, вписываясь в интерьер. Смотрите за тем чтобы объекты не были просто разбросаны.

3. **Настройка логики проекта в Blockly по примеру.**



4. **Сохранение логики и проверка работоспособности проектов в desktop режиме.**



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое Blockly?

2. Назовите несколько блоков из Blockly, с которыми Вам уже удалось поработать.
3. Какие основные отличия управления в Desktop и VR режиме.

Занятие 1.4 Создание макета города

Цель:

Усвоение навыков, полученных в ходе практических занятий. Разработка своего первого VR-мира.

Задачи:

- Сформировать понимание работы с кейсовыми заданиями
- Повысить навыки пространственного мышления
- Получить навыки рисования скетчей/ небольших планов локации
- Усвоить навык позиционирования объектов на сцене
- Научиться тестировать работоспособность собственных проектов

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

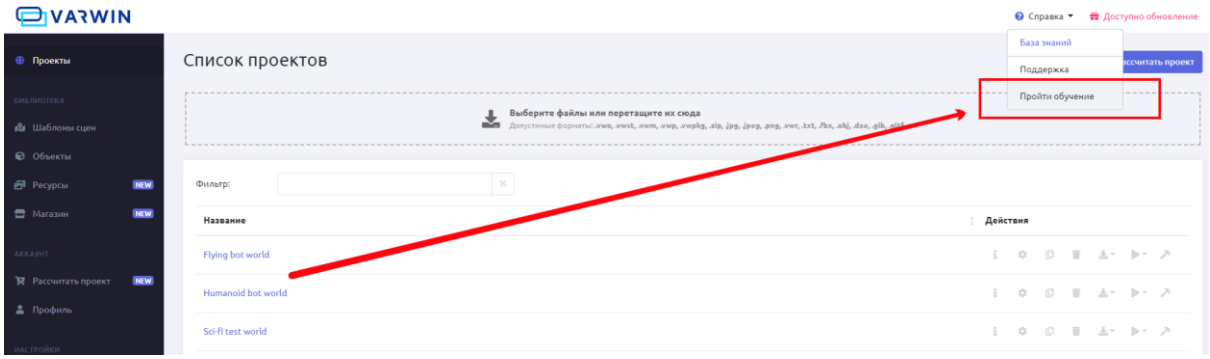
Формат: Самостоятельная работа, решение кейсового задания.

Ссылка на актуальную документацию Varwin:

Работа с Vive Focus — Документация Varwin 0.7.0 Beta

- Для **взаимодействия** с объектами нажмите **курок**.
- Для **телепортации** нажмите и удерживайте **сенсорную панель**, чтобы прицелиться и отпустите, чтобы телепортироваться.
- Для дискретного поворота вправо/влево нажимайте на правую/левую части сенсорной панели.
- Чтобы **взять** или **отпустить** объект, нажмите кнопку **«Приложения»**.
- Чтобы выйти из проекта на экран подключения, нажмите и удерживайте в течении двух секунд кнопку **«Приложения»**.
- Чтобы вызвать системное меню Vive Focus, нажмите кнопку **«Домой»**.
- Для взаимодействия с элементами интерфейса нажмите **сенсорную панель** или **курок**.

Если вы пользуетесь ПК-гарнитурой, то пройдите обучение.
Для этого нажмите на кнопку “**Справка**” и выберите пункт выпадающего меню “**Пройти обучение**”.



***Примечание**: если у вас нет VR-гарнитуры, не расстраивайтесь, очень скоро она будет у всех, как когда-то стал неотъемлемой вещью мобильный телефон. А пока пройдите созданный проект в режиме просмотра на ПК.*

Ссылка на актуальную документацию Varwin:
Использование VR контроллеров — Документация Varwin 0.7.0 Beta

10 - 20 минут	Получение кейса. Изучение объектов пакета “Мегаполис”. Формирование собственного скетч-плана города.
20 - 40 минут	Разработка проекта согласно скетч-плану.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

40 - 45 минут	Сохранение проектов. Тестирование и демонстрация проектов на VR-HMD устройствах.

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Кейс:
Построить небольшой макет города по собственному проекту.

Дополнительное задание, если позволяет время: применить простые логические конструкции в городе. Например, возможность включить фонари, используя объект “простая лампочка” и “простая кнопка”.

Обязательные условия:

1. Обязательно использовать как минимум 5 объектов пакета “Мегаполис” (другие пакеты объектов использовать НЕ запрещено)
2. Нарисовать скетч-план собственного города.
3. Эстетичность и правдоподобность расположения объектов на сцене.

2 Фото и видео 360

Занятие 2.1 Ресурсы и свойства объектов

Цель:

Сформировать понимание термина “Сферическая панорама” и познакомиться с первыми панорамами.

Задачи:

- Познакомить обучающихся с понятием “Панорама”
- Изучить разные типы панорам
- Получить первый опыт просмотра сферической панорамы в XRMS Varwin
- Научить обучающихся искать и скачивать из сети интернет сферические панорамы
- Сформировать понимание ресурсов в XRMS Varwin
- Изучить типы и требования к ресурсам, а также как их применять к объектам на сцене
- Изучить базовые объекты и их свойства в Desktop-редакторе из пакета “Простые объекты”, в т.ч. объект “Панорама”

Методические материалы для подготовки к занятию:

1. Пошаговая инструкция по разработке кейсов #2.
2. Что такое сферическая панорама?
3. Сферическая панорама

Литература для подготовки к занятию:
Что такое сферическая панорама?

Сферическая панорама (виртуальная панорама, 3D панорама) — один из видов панорамной фотографии. Предназначена в первую очередь для показа на компьютере (при помощи специального программного обеспечения).

В основе сферической панорамы лежит собранное из множества отдельных кадров изображение в сферической (эквидистантная, equirectangular, sphere) или кубической проекции. Характерной чертой сферических панорам является максимально возможный угол обзора пространства (360×180 градусов).

Литература для подготовки к занятию:
Сферическая панорама

Возможные ошибки при использовании обычных панорам:

Если вы используете обычную панорамы, а не сферическую или кубическую проекцию, то у Вас не получится при размещении ее в проекте заполнить всё пространство сферы и будет виден шов/переход в каком-то месте. Такие панорамы не подходят для наших проектов.

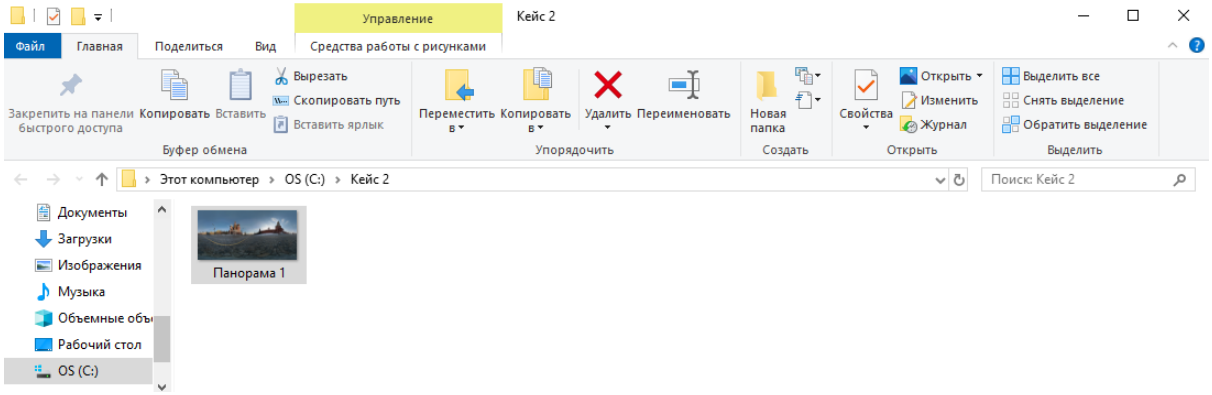
А теперь перейдем к XRMS Varwin и посмотрим как панорамы можно использовать там.

15 - 25 минут	Преподаватель демонстрирует экран персонального компьютера. Обучающиеся наблюдают и запоминают, что происходит на экране. Фронтальная работа.

Ресурсы Varwin

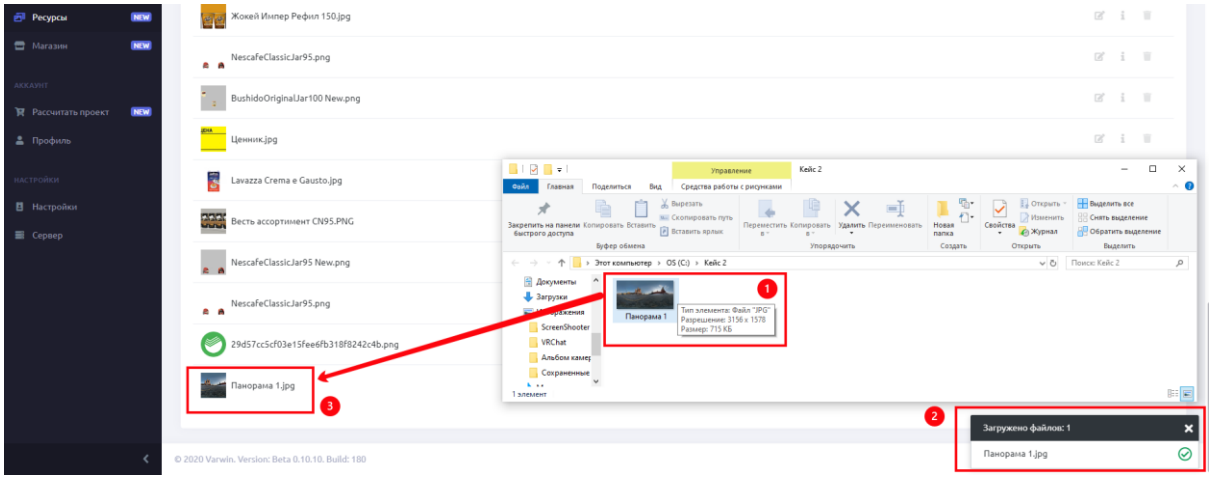
Сейчас нам надо понять что такое ресурсы и как их загружать в XRMS Varwin. Давайте на примере сферической панорамы попробуем это сделать. Итак у нас есть готовая сферическая панорама на нашем компьютере.

Мы загрузили панораму в необходимую директорию, в данном частном случае - это “C:\Кейс 2”.



После этого переходим в раздел “Ресурсы”, который находится в боковом меню Varwin и переносим методом “drag and drop” изображение в окно ресурсов (1).

После этого вы увидите всплывающее окно об успешной загрузке изображения в ресурсы Varwin (2). Панорама 360 отобразится в ресурсах (3).



Определение:

Ресурсы Varwin *- это мультимедиа файлы, распространенных форматов, которые можно загружать в платформу напрямую для дальнейшего использования в проектах.*

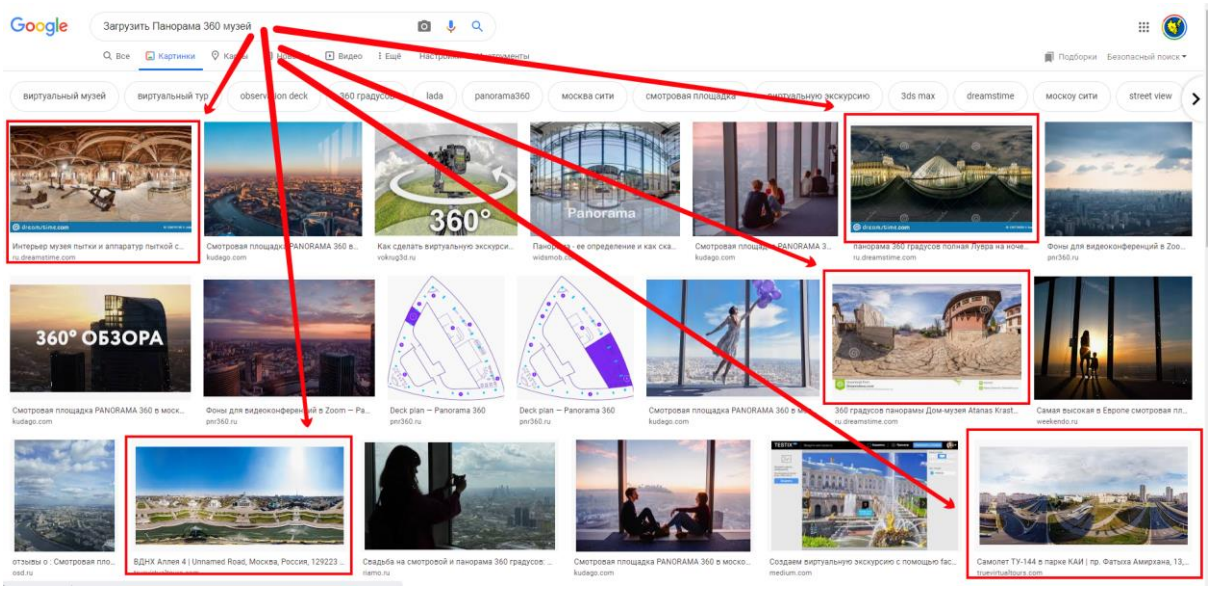
Примечание*: в качестве ресурсов Varwin могут выступать не только изображения, но и тестовые файлы, а также 3D-модели.*

Создание или поиск 360 панорам.

Если вы счастливый обладатель камеры, снимающей 360 панорамы, то вы можете создать собственные изображения.

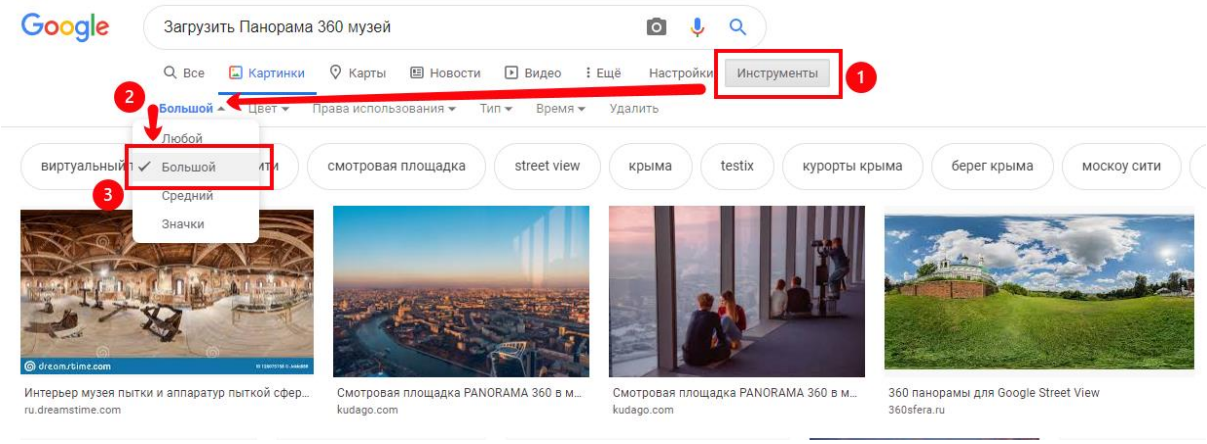
Но если такой камеры нет, не беда! Вы всегда сможете найти панорамы на необходимую тематику в поисковых системах.

Вот, например, результат поиска в **Google-картинках** по запросу **“Загрузить Панорама 360 музей”**:



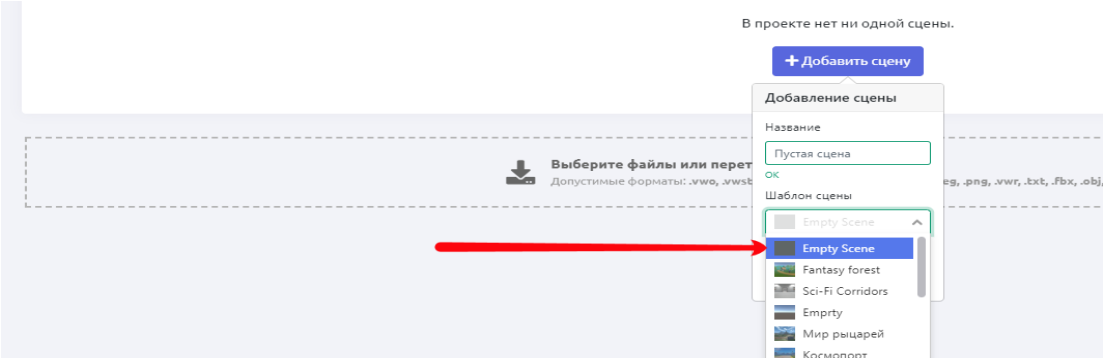
Вы сразу же сможете различить необходимые 360 панорамы по характерной композиции, как на картинке выше. Сохраните их в максимально возможном качестве.

Для этого вы можете отфильтровать панорамы в самом высоком разрешении по вашему запросу, достаточно нажать кнопку “Инструменты” (1) → “Размер” (2) → “Большой” (3).



Вспомним первые шаги.

Для начала, как мы это делали в предыдущем модуле, создайте новый проект и добавьте сцену “Empty Scene” (рус. “Пустая Сцена”).

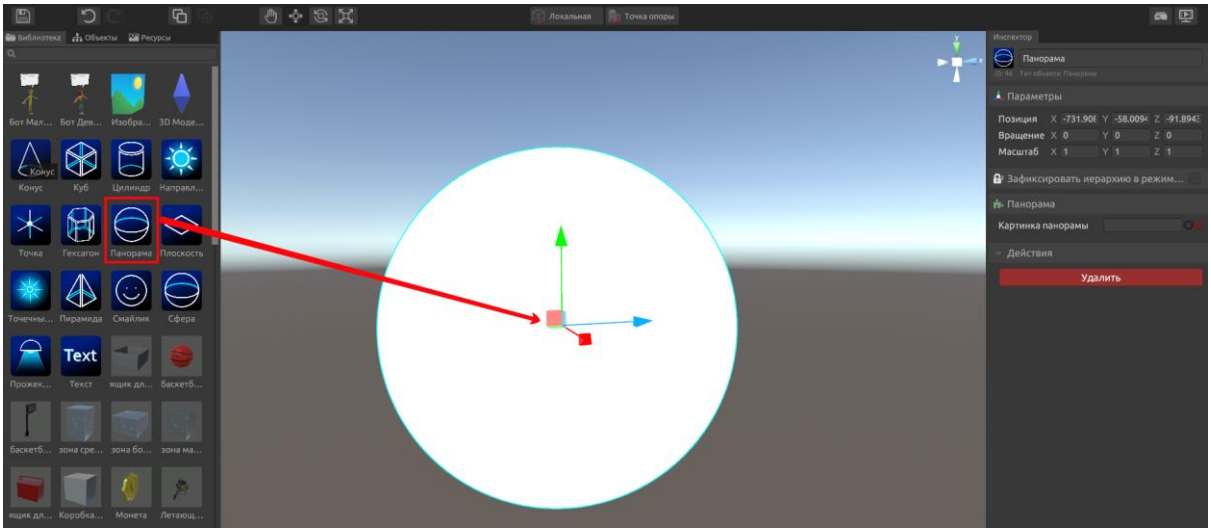


***Примечание**: “Empty Scene” пригодится, если вы хотите создать все окружение “с нуля”.*

Далее открываем редактор сцен (Desktop-редактор).

Создание панорамы 360.

Добавляем в любую позицию на сцене объект “Панорама”.

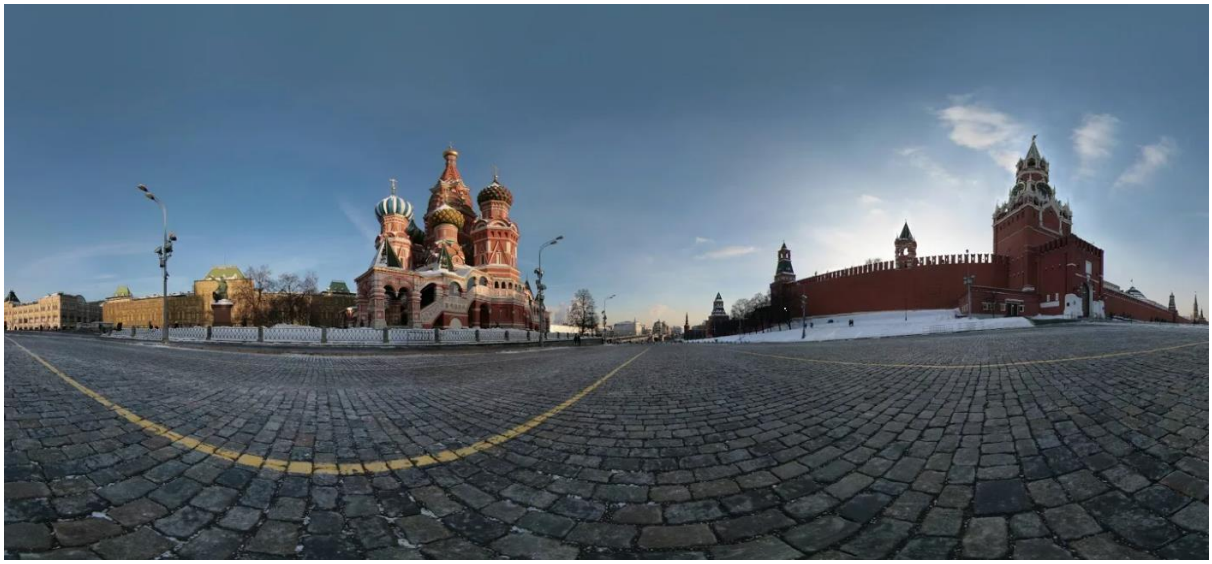


На объект панорама мы можем применить изображение в формате 3D панорама, например, для создания экскурсий. Но для начала давайте загрузим это изображение.

***Определение**:*

***3D панорама или панорама 360°** - это интерактивное фотоизображение позволяющее показать окружающее пространство вокруг точки съемки со всех сторон. На плоскости 3D панорама можете быть*

представлена в виде эквидистантной проекции, так называемая равноугольная развертка.



Это может быть не обязательно фото, снятое на специальную аппаратуру, но и результат рендеринга 3D-сцены.

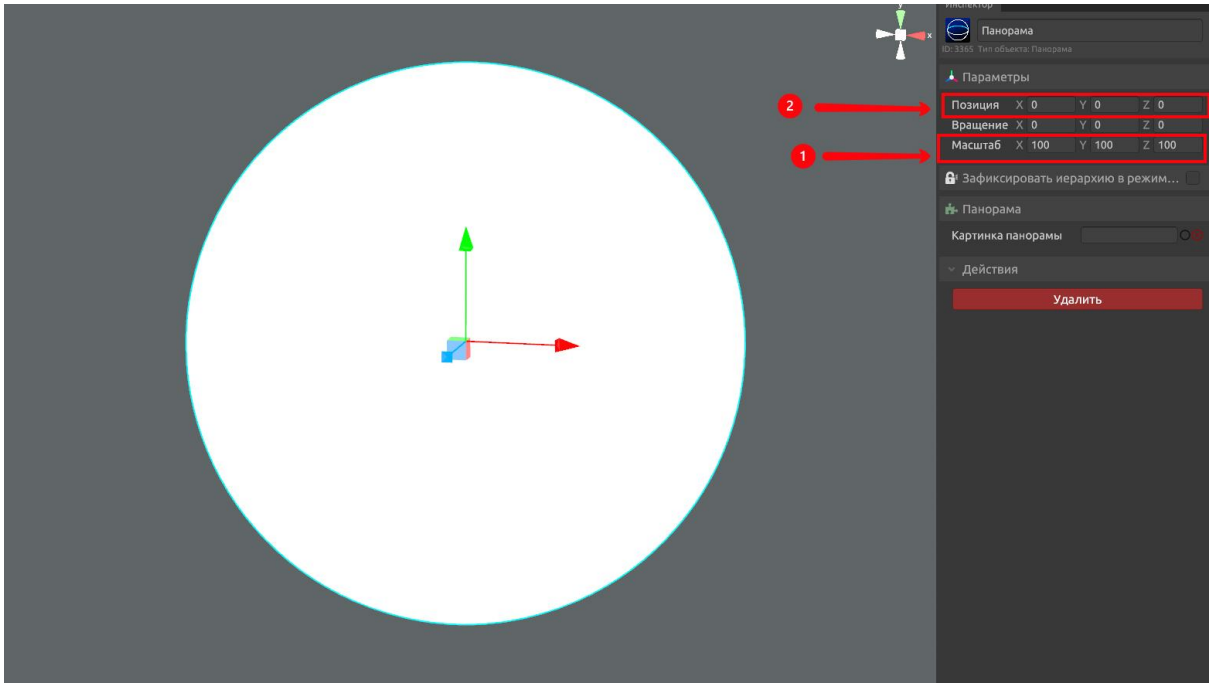
***Определение**:*

***Рендер (Рендеринг)** — это процесс создания финального изображения или последовательности из изображений на основе двухмерных или трехмерных данных. Данный процесс происходит с использованием компьютерных программ и зачастую сопровождается трудными техническими вычислениями, которые ложатся на вычислительные мощности компьютера или на отдельные его комплектующие части.*

Выбор ресурса для панорамы

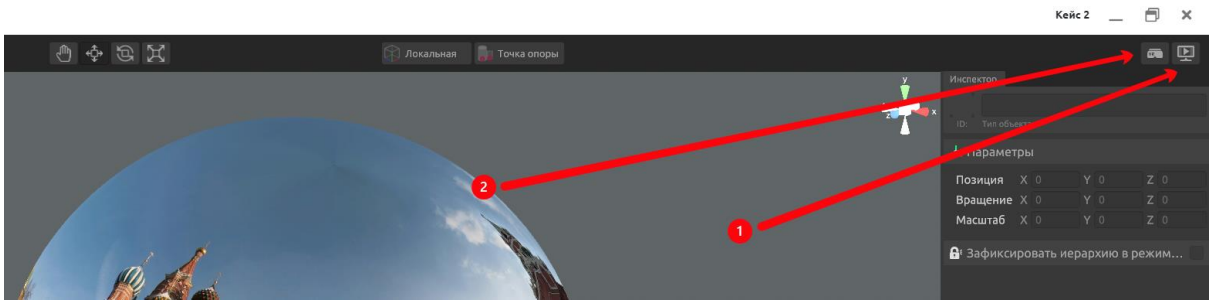
Вернемся в Desktop-редактор и выделим нашу панораму. В текущий момент она имеет очень маленький масштаб, несоразмерный игроку.

- Увеличьте масштаб панорамы в 10 раз, пропорционально всем осям (1).
- Установите позицию в координатах “0; 0; 0”, чтобы игрок появлялся ровно в центре панорамы 360 (2).



- Нажмите на черную окружность для выбора ресурса панорамы (1).
- Выберите в списке ресурсов необходимую панораму - панорама применяется к объекту Varwin.

Теперь можете запустить проект прямо из Desktop-редактора в **режиме просмотра на ПК** (1) (или, если вы смелый и у вас подключен шлем виртуальной реальности, то сразу в **режиме VR** (2)).



Осмотритесь, теперь вы точно знаете, какое погружение дает 360 панорама по сравнению с плоской фотографией.

***Самостоятельное задание**: создайте таким же методом минимум еще 2 панорамы.*

Контрольные вопросы:

1. Что такое сферическая панорама?
2. В какой панели вносят изменения размеров и расположения объектов? Например, объекта “Панорама”?
3. Назовите основные требования к ресурсу “Панорама” для XRMS Varwin.
4. Что такое рендер/рендеринг?

Занятие 2.2 Создание VR-экскурсии

Цель:

Разработать проект VR-экскурсии и протестировать его в VR.

Задачи:

- Научить обучающихся размещать несколько сферических панорам на сцене
- Сформировать понимание пользовательского интерфейса приложения
- Научить обучающихся создавать пользовательский интерфейс, в т.ч. кнопки для перемещения между панорамами
- Усвоить навык тестирования работоспособности собственных проектов
- Изучить основные свойства объектов и их применение
- Научиться работать с простой логикой событий в Blockly и точками появлений игрока на сцене

Методические материалы для подготовки к занятию:
Пошаговая инструкция по разработке кейсов #2.

Создание точек появления игрока.

Определение:
Точки появления игрока - это объект Varwin, который определяет положение игрока на сцене в зависимости от условий.

Вы уже могли обратить внимание, что по умолчанию на сцене всегда находится как минимум одна **точка появления игрока** и сам **игрок**. Без этого проект просто не будет работать, т.к. игрок и его точка появления - это базовые объекты, без которых запуск проекта просто не имел бы смысла.

***Вопрос**: почему без игрока запуск проекта не имеет смысла?*

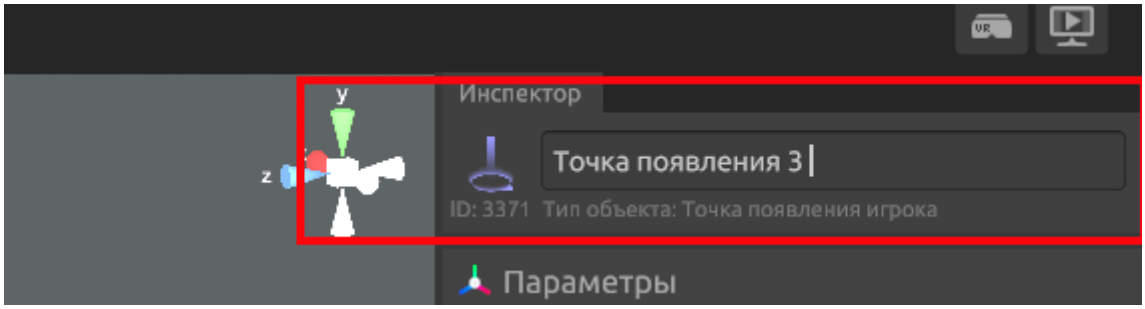
На данном занятии нас интересует перемещение игрока в центр различных панорам, в зависимости от условий. Условия мы определим позже, а

для начала давайте разместим эти точки.Для этого удобнее всего использовать метод ввода координат вручную.

Алгоритм прост:

1. Добавляем на сцену **X** точек появления игрока = **X** количеству панорам.
2. Присваиваем параметры позиции точки появления игрока равные соответствующей панораме.

***Совет**: Давайте будем вводить культуру наименований. Чтобы не запутаться, например, у "Панорамы 2" будет "Точка появления 2", у "Панорамы 3" → "Точка появления 3". Поверьте, это очень сильно пригодится нам при создании логики взаимодействия. Вводить наименования можно в инспекторе, не забудьте нажать клавишу "Enter" после ввода.*



- Создайте **точки появления игрока** в центре каждой **панорамы**.

Создание пользовательских интерфейсов перехода между панорамами

Следующим шагом вам необходимо создать механизм перехода между панорамами с помощью пользовательских интерфейсов.

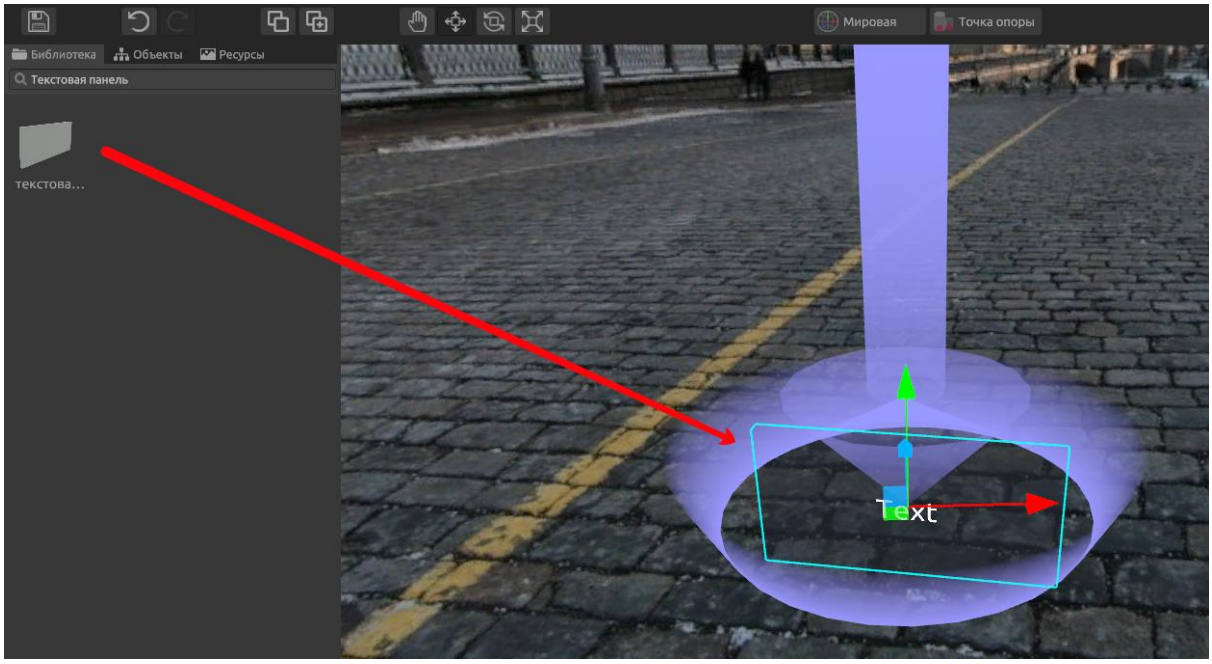
Определение:
***Пользовательские интерфейсы или UI (англ. "user interface")** - это графическая оболочка. Кнопки, на которые жмёт пользователь, тексты, которые читает, изображения, формы ввода данных и другие интерактивные элементы.*

В данном проекте все панорамы у нас расположены друг за другом, поэтому по факту нам понадобится два типа UI:

1. Переход к следующей панораме
2. Переход к предыдущей панораме

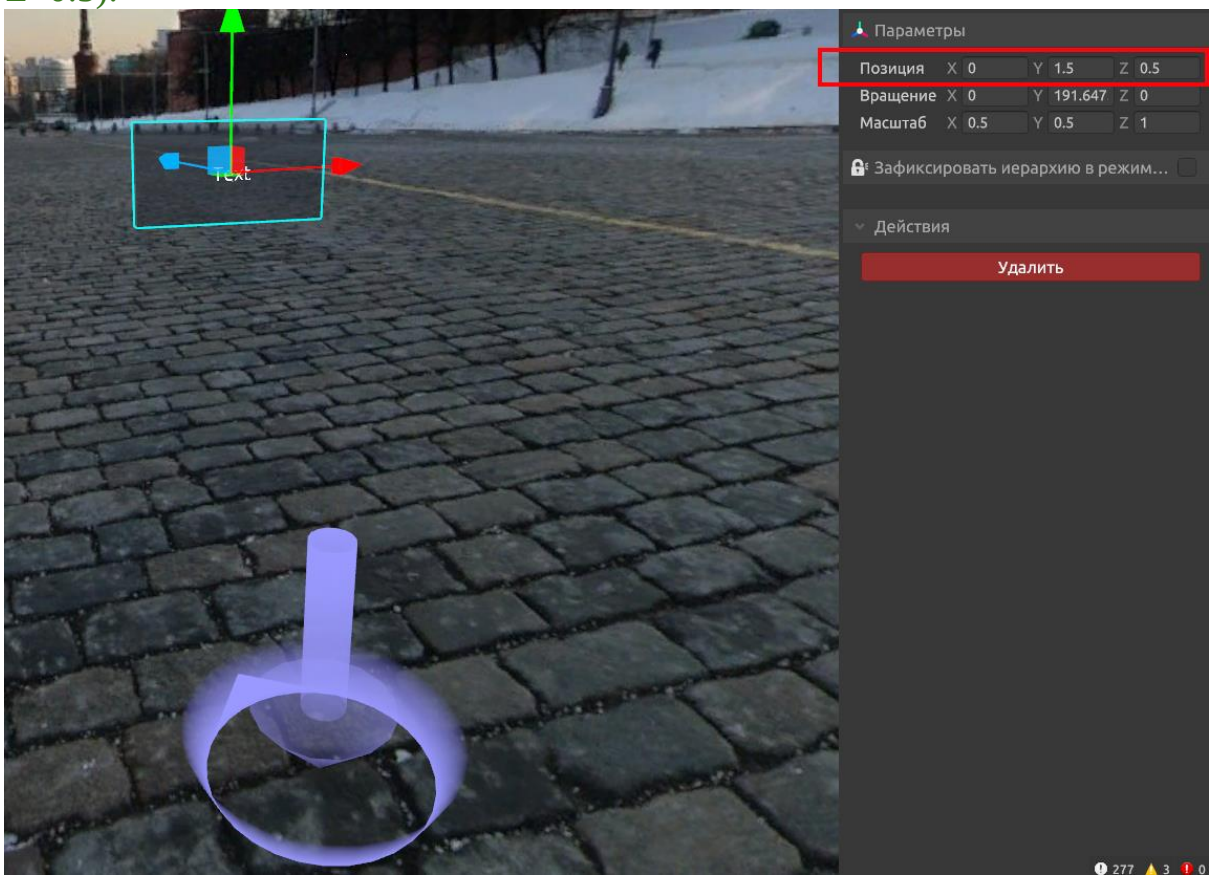
Для начальной и конечной панорамы будет только один тип UI, т.к. они находятся на концах нашего пути следования. Логично начать создание переходов с самой первой панорамы, т.е. с начала.

Для создания UI удобно будет использовать объект Varwin "**Текстовая панель**". Давайте для начала разместим этот объект с позицией **точки появления игрока по умолчанию**, т.е. "0, 0, 0".



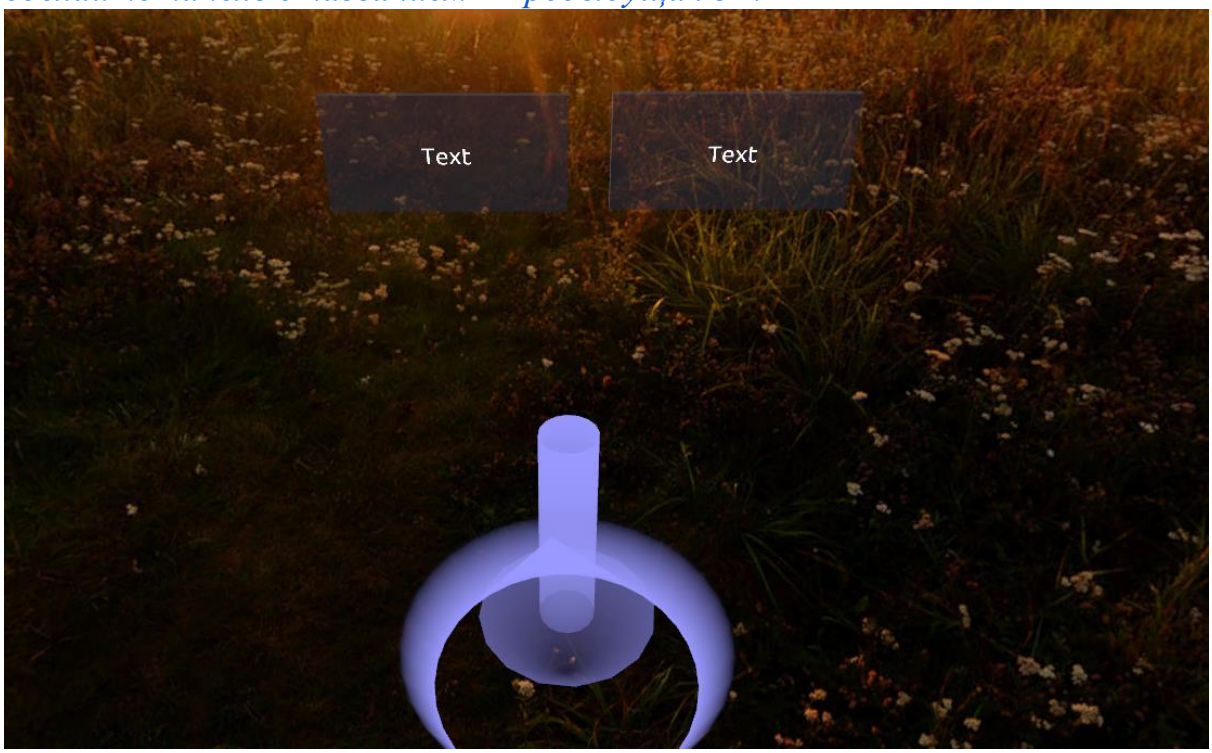
Чтобы игроку было удобно использовать UI, его необходимо правильно расположить относительно точки появления игрока.

В данном случае предлагаются координаты, на картинке ниже (X=0; Y=1.5; Z=0.5):



Вы всегда можете проверить удобство расположения UI запустив проект в различных режимах. Делайте это на каждом этапе разработки.

Самостоятельная работа: *Для второй точки сделайте UI примерно как на картинке ниже. Дайте наименование UI “Следующая 2” и “Предыдущая 2”. Для третьей точки (если она последняя в вашем случае), сделайте панель с названием “Предыдущая 3”.*

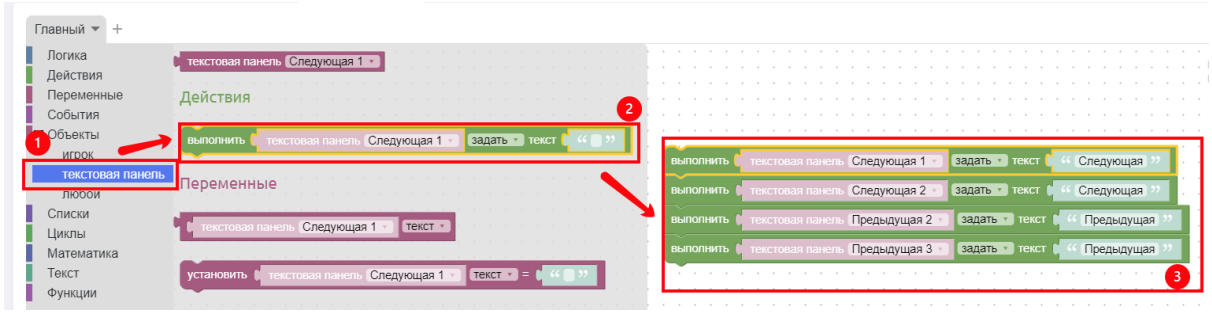


Создание логики перехода

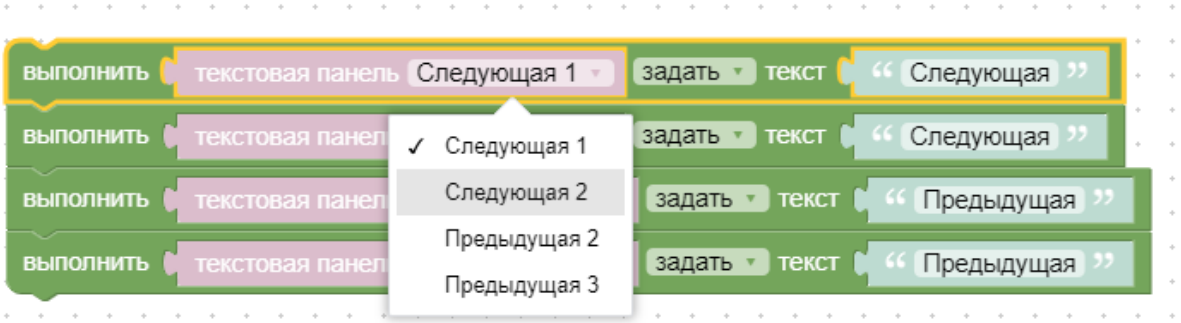
Мы расположили все необходимые объекты на сцене, пришло время создать логику. Для этого откроем редактор логики.

Установка текста

Для начала зададим текст на текстовых панелях, в зависимости от их назначения.



Текстовая панель это один тип объекта. На сцене может быть множество объектов одного типа, и в редакторе логики в категории “Объекты” будут отображаться как один тип. Вам необходимо выбрать конкретное наименование объекта из одного типа, чтобы задать логику именно ему, как на картинке ниже:



Здесь очень просто запутаться, если не именовать объекты. Теперь вы понимаете почему важно грамотно задавать имена объектам Varwin, особенно, если они одинакового типа.

Создание логики перехода через события

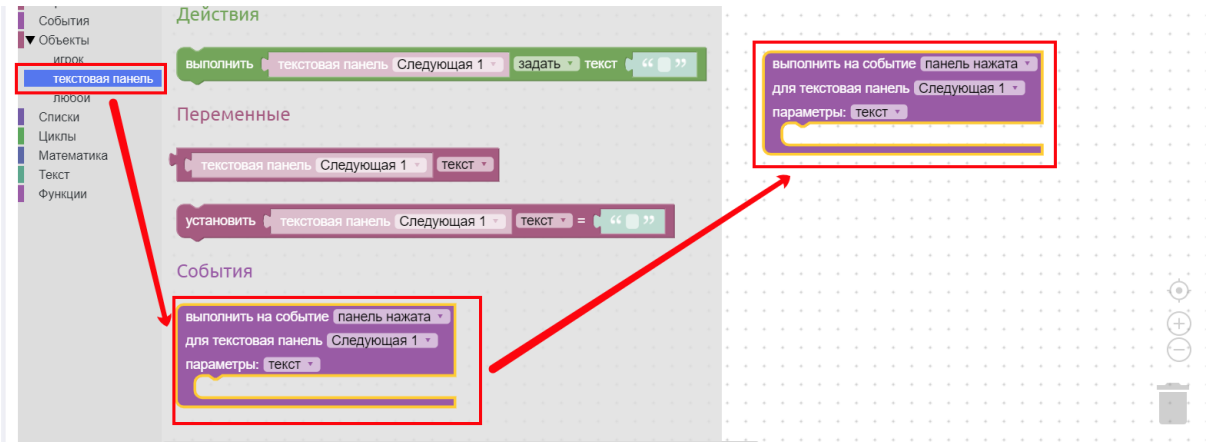
Дальше нам необходимо задать логику перехода между панорамами.

Для этого давайте сформируем **техническое задание**. Сейчас оно может выглядеть элементарно и слишком наивно, но здесь важно понимать - правильно сформированное техническое задание - это залог хорошего проекта.

В разработке дальнейших, более сложных, проектов возьмем за правило начинать с формирования технического задания (сокращенно “**ТЗ**”).

ТЗ: при нажатии на UI “Следующая” мы перемещаемся к следующей панораме. При нажатии на UI “Предыдущая” мы возвращаемся к предыдущей панорме.

Для этого выберите и перетащите в рабочую область блок “Текстовой панели” - Событие “Панель нажата”.

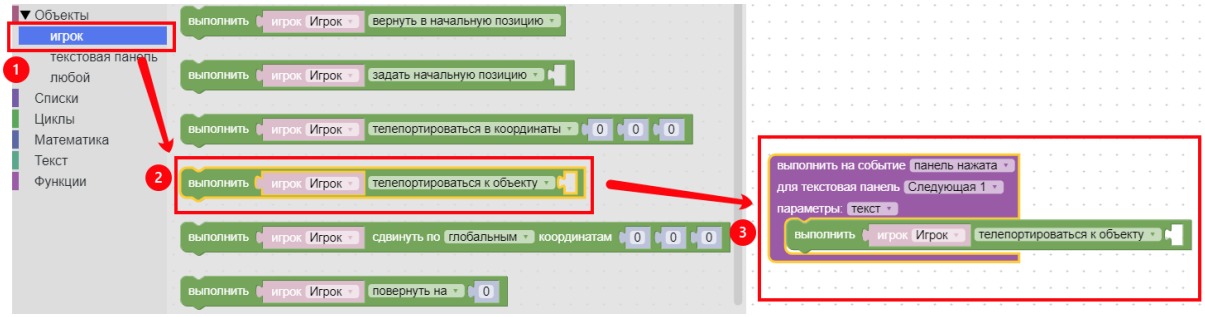


Определение:

Событие *- это логика, которая инициализируется при совершении определенных действий игроком. Целевое действие обозначено в условиях блока. В нашем случае событие инициализируется при совершении целевого действия - использование текстовой панели.*

При активации данного события мы должны переместить игрока в **точку появления 2**, которая соответствует **панораме 2**. Для этого выберем в редакторе логики в категории “Объекты” тип “Игрок” (1). Нам понадобится

блок ”Телепортировать к объекту” (2). Вставим этот блок в логику, которая будет реализовываться при активации события (3).

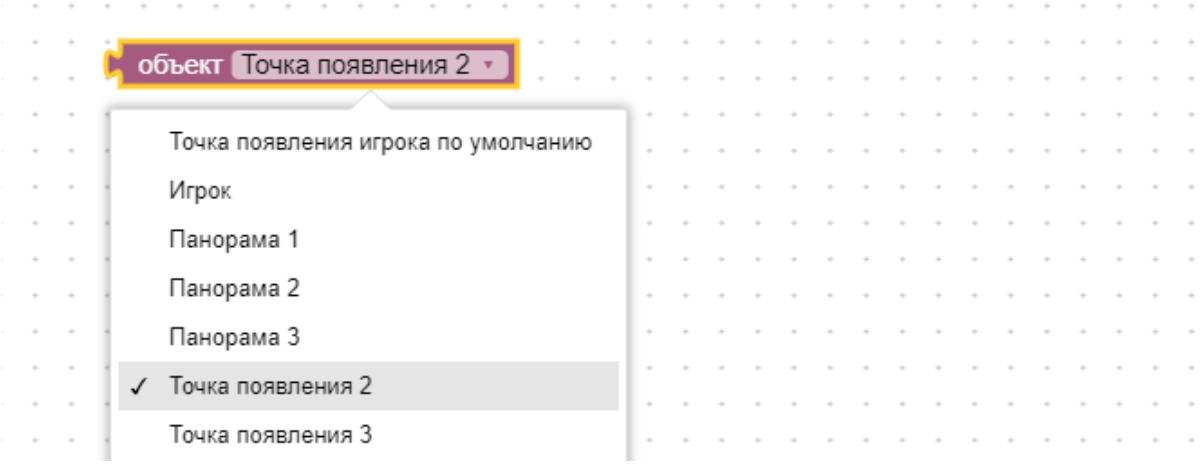


Последнее, что нам нужно сделать, это определить к какому объекту будет телепортироваться игрок. Вы помните, что это точки появления игрока, но точка появления игрока не появляется в редакторе логики как отдельный тип в категории объектов.

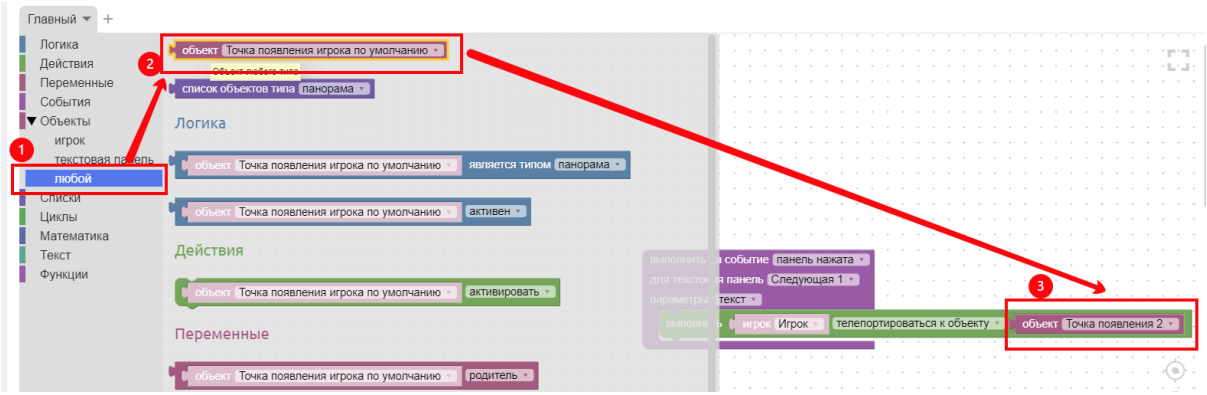
Это хороший пример того, как найти любой объект на сцене, даже если он не выделен в отдельный тип.

Для этого выберем тип “Любой” в категории “Объекты” (1).

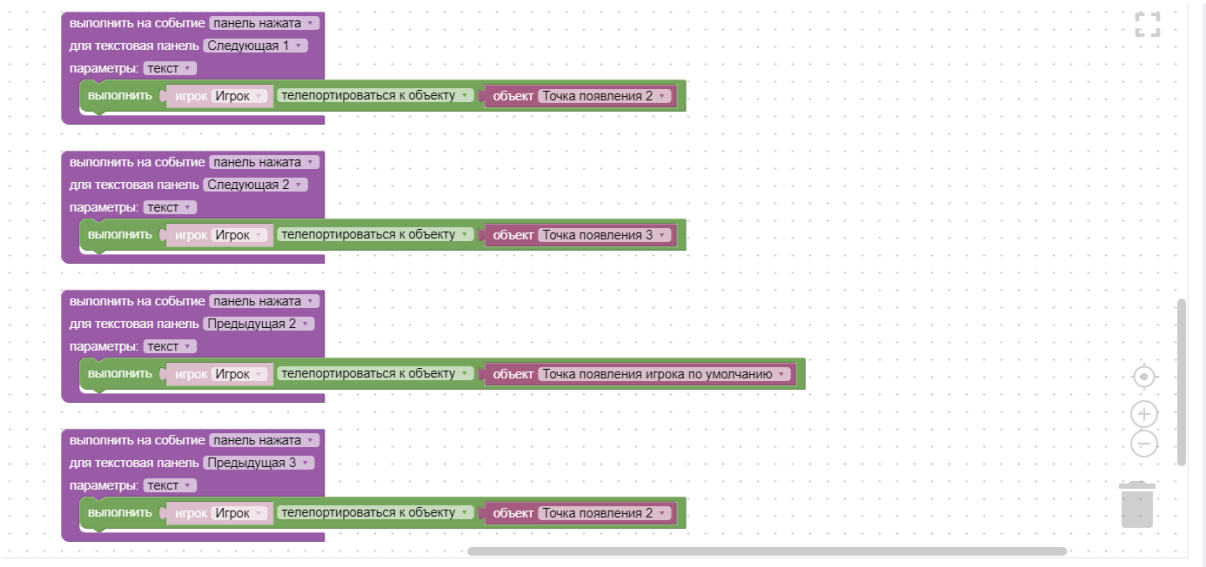
Далее выберем блок “Объект” и из выпадающего меню выберем необходимое имя (2).



Вставим блок объекта в логику выполнения события (3).



Самостоятельная работа: *Проделайте похожую операцию для остальных событий, чтобы получилась следующая логика (актуально для трех панорам):*



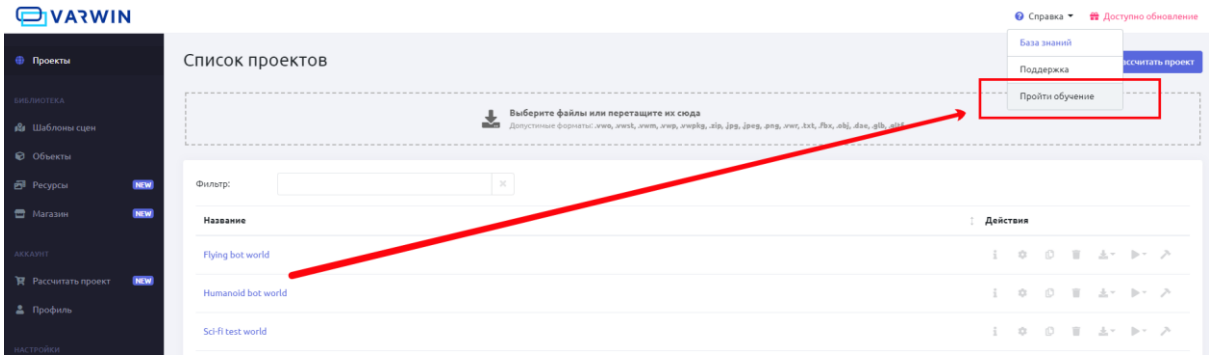
Запуск проекта.

Попробуйте запустить проект в режиме VR для максимального погружения. Если вы используете Vive Focus, то следуйте инструкции:

Ссылка на актуальную документацию Varwin:
Работа с Vive Focus — Документация Varwin 0.7.0 Beta

- Для **взаимодействия** с объектами нажмите **курок**.
- Для **телепортации** нажмите и удерживайте **сенсорную панель**, чтобы прицелиться и отпустите, чтобы телепортироваться.
- Для дискретного поворота вправо/влево нажимайте на правую/левую части сенсорной панели.
- Чтобы **взять** или **отпустить** объект, нажмите кнопку **«Приложения»**.
- Чтобы выйти из проекта на экран подключения, нажмите и удерживайте в течении двух секунд кнопку **«Приложения»**.
- Чтобы вызвать системное меню Vive Focus, нажмите кнопку **«Домой»**.
- Для взаимодействия с элементами интерфейса нажмите **сенсорную панель** или **курок**.

Если вы пользуетесь ПК-гарнитурой, то пройдите обучение.
Для этого нажмите на кнопку “**Справка**” и выберите пункт выпадающего меню “**Пройти обучение**”.



***Примечание**: если у вас нет VR-гарнитуры, не расстраивайтесь, очень скоро она будет у всех, как когда-то стал неотъемлемой вещью мобильный телефон. А пока пройдите созданный проект в режиме просмотра на ПК.*

Ссылка на актуальную документацию Varwin:
Использование VR контроллеров — Документация Varwin 0.7.0 Beta

Если у вас получилось сделать экскурсию из трех панорам с правильными переходами от следующей к предыдущей и обратно, то вы справились! Поздравляю!

40 - 45 минут	Учебная дискуссия. Групповая работа.

Рефлексия. Обсуждение у всех ли получилось настроить логику последовательных переходов. Понравилось ли Вам создавать панорамные экскурсии? Готовы ли Вы сами реализовать полноценную VR-экскурсию, есть идеи экскурсий?

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое пользовательский интерфейс (user interface)?
2. Назовите идеальный рецепт количества точек появления игрока и количества панорам.
3. Что максимально важно делать при работе с большим количеством объектов на сцене?
4. Что такое “Событие”?

Занятие 2.3 Разветвленная экскурсия

Цель:

Усвоение навыков, полученных в ходе практических занятий. Разработка разветвленной VR-экскурсии по собственному техническому заданию.

Задачи:

- Повысить навыки пространственного мышления

- Получить навыки рисования скетчей/ планов перемещения по виртуальному пространству
- Усвоить навык работы с сферическими панорамами
- Усвоить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Научить обучающихся формировать и фиксировать техническое задание (далее ТЗ) проекта
- Усвоить базовые навыки работы с логикой в Blockly
- Закрепить навыки работы с desktop - редактором XRMS Varwin

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

Формат: Самостоятельная работа, решение кейсового задания.

Методические материалы для подготовки к занятию:

1. Пошаговая инструкция по разработке кейсов #2.
2. Техническое задание
3. Soft-skills

На этом этапе нужно научить обучающихся формировать и фиксировать техническое задание. Дать им понять что это такое:

Литература для подготовки к занятию:

Техническое задание (ТЗ, техзадание) — документ, содержащий требования заказчика к объекту закупки, определяющие условия и порядок ее проведения для обеспечения государственных или муниципальных нужд, в соответствии с которым осуществляются поставка товара, выполнение работ, оказание услуг и их приемка.

Также очень важно зарисовать план перемещения по виртуальному пространству. То есть в каких направлениях может двигаться игрок.

Шаблон технического задания:

1. Тематика/направленность проекта
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы)
3. Дополнительные возможности
4. План-схема сцен, которые будут реализованы

Совет: четко и грамотно модерируйте время на формирование собственного технического задания. Обучающиеся могут сильно увлечься и расплывчато формировать ТЗ. Здесь нужно четко и конкретно (желательно тезисами) сформировать ТЗ и нарисовать структурный план переходов между панорамами.

10 - 40 минут	Разработка проекта согласно техническому заданию.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

40 - 45 минут	Сохранение проектов. Тестирование и демонстрация проектов на VR-HMD устройствах.

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Определение:

Ги́бкие на́выки или **мя́гкие на́выки** (англ. *soft skills*) — комплекс неспециализированных, важных для карьеры надпрофессиональных навыков, которые отвечают за успешное участие в рабочем процессе, высокую производительность и являются сквозными, то есть не связаны с конкретной предметной областью.

Кейс:

Разработать разветвленную экскурсию в виртуальном пространстве используя сферические панорамы и элементы UI-дизайна.

Дополнительное задание, если позволяет время: реализовать дополнительные логические конструкции, которые должны быть зафиксированы в ТЗ. Например, возврат в начальную панораму из любой точки приложения.

Обязательные условия:

1. Зарисовать план перемещения по виртуальному пространству
2. Сформировать и зафиксировать технического задание проекта
3. Создать более сложную сетку перемещения между панорамами, с тремя направлениями и более

Использовать только качественные сферические панорамы хорошего разрешения

3 Условные операторы. Переменные

Занятие 3.1 Условные операторы

Цель:

Изучить условные операторы, понять их назначение и применяемость в наших проектах

Задачи:

- Закрепить знание ресурсов XRMS Varwin
- Усвоить навык изменения свойств объектов на сцене
- Познакомиться с условными операторами
- Познакомиться с основами UX/UI дизайна
- Сформировать понимание UI дизайна для проектов на XRMS Varwin

Методические материалы для подготовки к занятию:

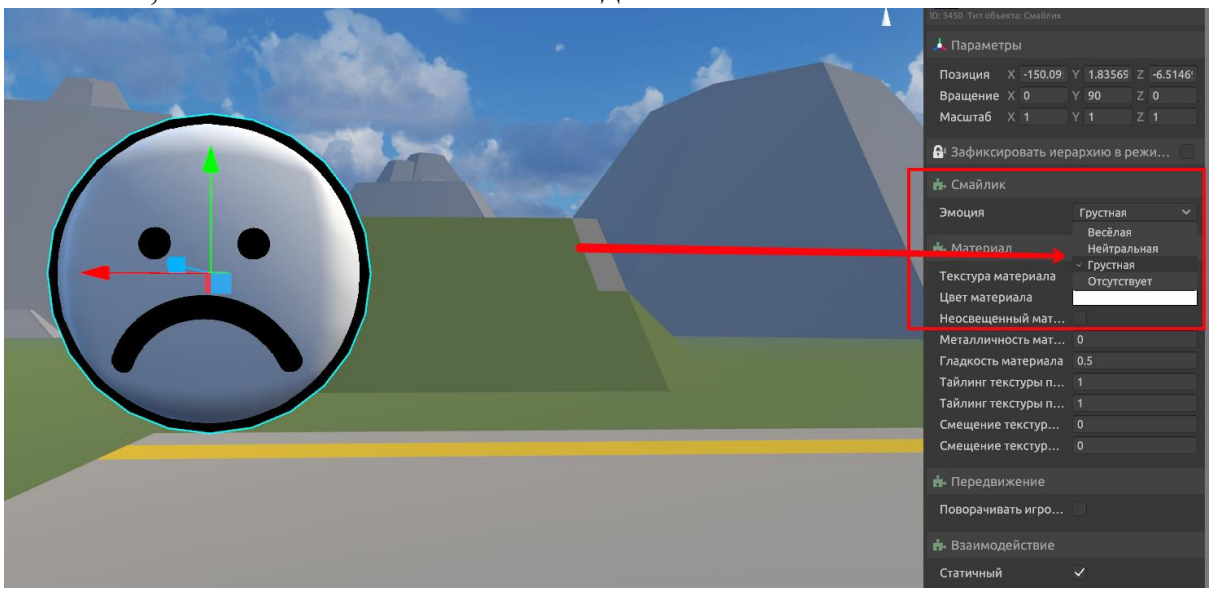
1. Пошаговая инструкция по разработке кейсов #3.
2. Условные операторы:
   a. Условие (программирование)
   b. Программирование условий: условный оператор, оператор выбора, Условный оператор if - Информатика
   c. Ветвление (программирование)
   d. Урок информатики "Составные условия в алгоритме ветвления"
3. UX/UI - дизайн:
   a. Принципы и основы UX-дизайна
   b. 9 шагов как стать UX/UI дизайнером, если нет образования и опыта работы.
   c. 10 золотых правил UI дизайна | UXPUB
   d. Дизайн интерфейсов: 10 правил UI-дизайна | DENISOV
   e. С чего начать обучение UX-дизайну и как стать профессионалом

Ресурсы Varwin

На этом этапе нужно уточнить что у разных объектов бывают разные свойства в окне инспектора.

Например, смайлик в зависимости от условий может менять свою эмоцию: безразличие, грусть, радость.

Также нужно уделить внимание подробному разбору основных свойств объектов, чтобы понимать за что каждый из них отвечает.



Определение:

Ресурсы Varwin *- это мультимедиа файлы, распространенных форматов, которые можно загружать в платформу напрямую для дальнейшего использования в проектах.*

***Примечание**: в качестве ресурсов Varwin могут выступать не только изображения, но и тестовые файлы, а также 3D-модели. И у каждого из них могут быть разные или по-разному настроенные свойства.*

В этой части лекционного занятия нужно сформировать понимание условных операторов в программировании. Нужно уделить внимание следующим операторам: **if do; else; else if (или elif).** Разобрать примеры с множественными условиями. И уделить время операторам сравнения “больше”, “меньше”, “равно”, “не равно”, “больше или равно”, “меньше или равно”. Также понять назначение операторов “&” (and) и “|” (or). Изучить булевы переменные (true, false).

Литература для подготовки к занятию:

Условие (программирование)

Литература для подготовки к занятию:

Программирование условий: условный оператор, оператор выбора, Условный оператор if - Информатика

Определение:

Опера́тор ветвле́ния (усло́вная инстру́кция, усло́вный опера́тор) — оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

Литература для подготовки к занятию:

Ветвление (программирование)

Литература для подготовки к занятию:

Урок информатики "Составные условия в алгоритме ветвления"

На следующем этапе нужно изучить основы UX/UI дизайна. Чтобы в будущем, основываясь на простых советах, обучающиеся смогли разработать качественный дизайн интерфейсов.

Определение: UX-дизайнер — это проектировщик, который делает цифровые продукты понятными и логичными. Он изучает опыт взаимодействия пользователя с сайтом, приложением или программой.

Цель работы UX-дизайнера — помочь пользователю максимально комфортно достичь своей цели, какой бы она ни была.

Литература для подготовки к занятию:

1. Принципы и основы UX-дизайна
2. 9 шагов как стать UX/UI дизайнером, если нет образования и опыта работы.
3. 10 золотых правил UI дизайна | UXPUB
4. Дизайн интерфейсов: 10 правил UI-дизайна | DENISOV
5. С чего начать обучение UX-дизайну и как стать профессионалом

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какая самая главная цель UX/UI-дизайнера?

2. Какие условные конструкции вы знаете и как их можно применять?
3. Какие блоки присутствуют в категории «Логика»? За то они отвечают?
4. Какие свойства есть у объекта? За что они отвечают?

Занятие 3.2 Зоны и продвинутые свойства объектов

Цель:

Разработать сцену проекта “Планеты” и подготовить ее для применения логических конструкций.

Задачи:

- Познакомиться с вспомогательным объектом “Зона”
- Изучить возможные логические конструкции для применения операторов условия
- Усвоить понимание UX/UI-дизайна и для чего он нужен
- Создать объекты для реализации комфортного и удобного UX/UI-дизайна
- Усвоить работу с базовыми свойствами объекта в инспекторе
- Изучить продвинутые свойства объектов объектов
- Расставить все необходимые объекты на сцене через desktop/vr редактор
- Закрепить навыки, полученные в лекционном материале.
- Усвоить навык работы по сформированному техническому заданию

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #3.

В кейсовом задании “Планеты” мы познакомимся с логическими конструкциями Blockly с применением условных операторов из категории “Логика” и познакомимся с базовыми свойствами объектов в инспекторе.

Для начала создайте новый проект, дайте ему название и выберите сцену. В примере будет сцена **“Космопорт”**, но вы можете выбрать альтернативную, главное условие - на сцене должен быть жесткий **“пол”** для перемещения, поэтому, например, **“Empty Scene”**, не подойдет.

Примечание: *в будущем мы познакомимся с приемом как можно сделать такой* ***“пол”*** *(или, если давать правильный термин -* ***“зона для телепортации”****) из любого объекта.*

Формирование Технического Задания

Для того, чтобы определиться какие объекты нам необходимо разместить на сцене, как мы договорились в предыдущей главе, начнем с формирования **Технического задания (ТЗ)**.

ТЗ: Необходимо создать проект на Varwin, который позволял бы проверять знания о порядке расположения планет солнечной системы, относительно солнца.

Обязательные требования:

1. Необходимо использовать минимум 3 первых планеты от Солнца.
2. В приложении должен быть реализован удобный UX/UI-дизайн (что это такое, поговорим, когда дойдем до этого шага), позволяющий пользователю понять что ему необходимо сделать и в какие точки размещать планеты.
3. В приложении должна присутствовать механика, позволяющая определить на правильное место мы поставили планету или нет.
4. После размещения всех планет на своих местах должно произойти событие, которое явно бы показывало пользователю о том, что он все сделал правильно.

Далее при разработке кейсового задания мы будем возвращаться к этим пунктам.
Пойдем по порядку.

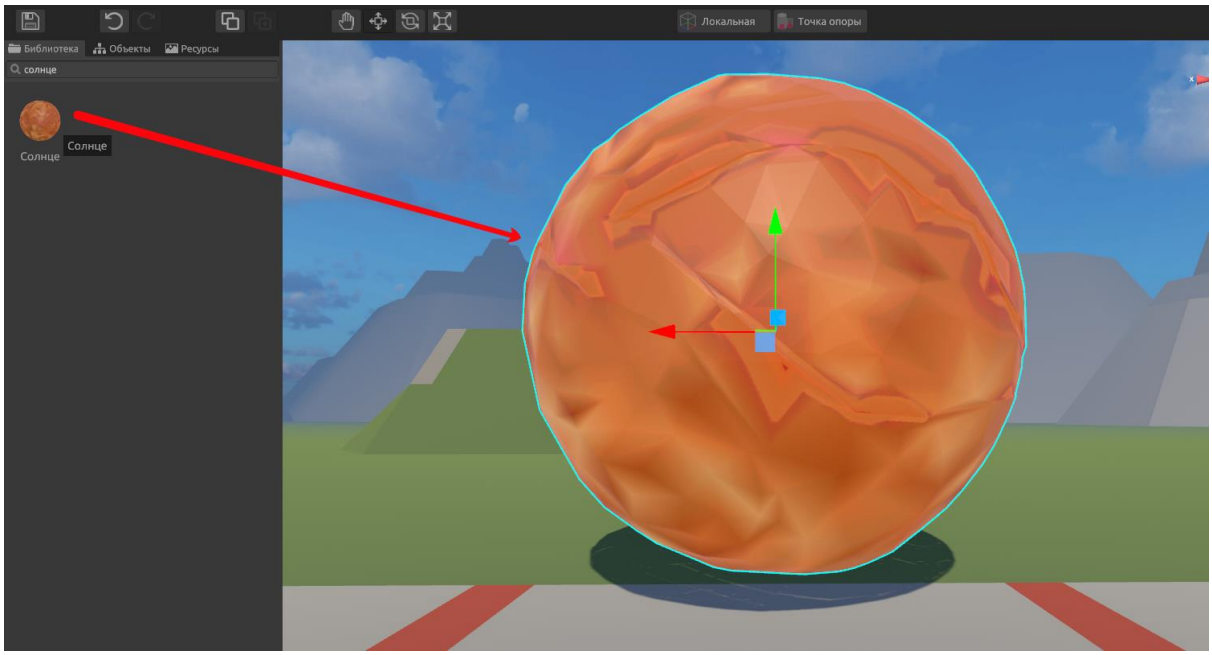
Размещение необходимых объектов на сцене

Сначала давайте еще раз почитаем ТЗ и поймем, какие объекты на сцене нам потребуются.

Размещение планет

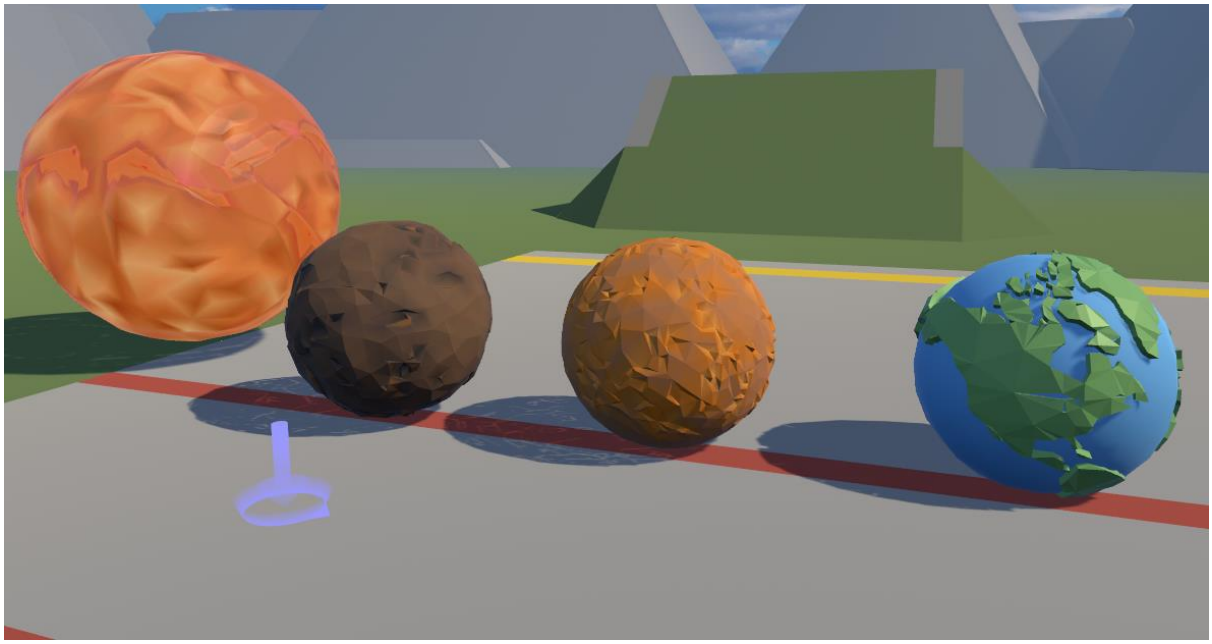
из ТЗ: *“Необходимо создать проект на Varwin, который позволял бы проверять знания о порядке расположения планет солнечной системы, относительно солнца”.*

Первый вывод, который можно сделать - нам понадобится Солнце, разместим его на сцене в удобном месте:



из ТЗ: *“Необходимо использовать минимум 3 первых планеты от Солнца”.*

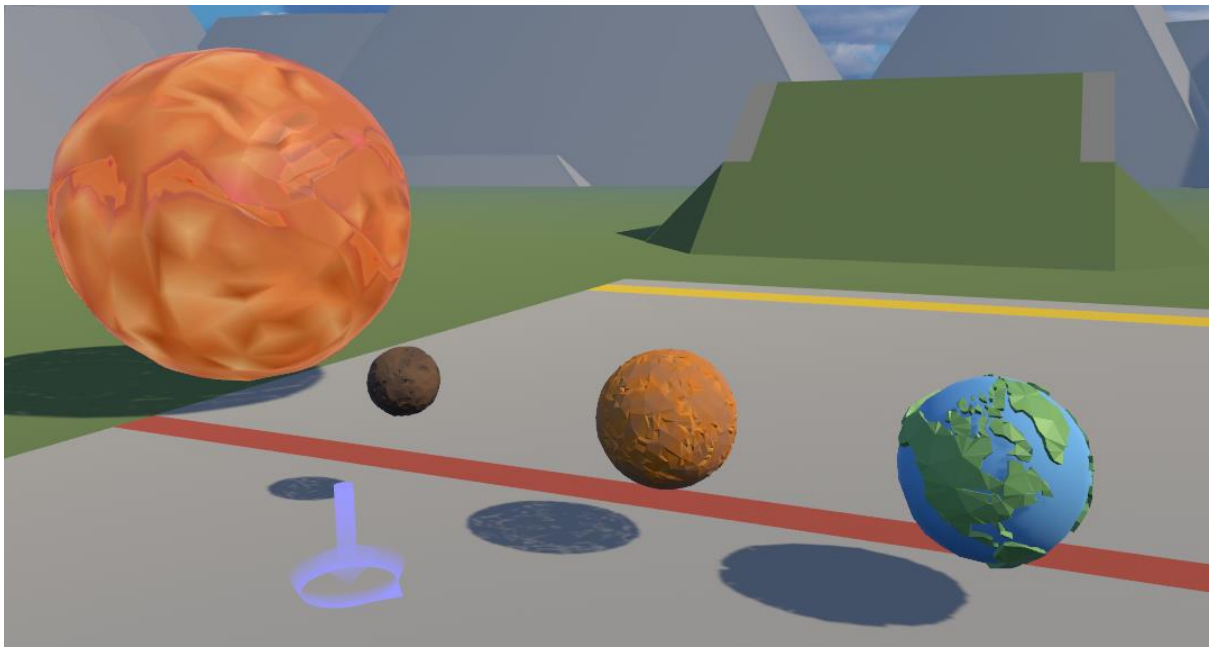
Кроме Солнца на понадобятся три первые планеты: Меркурий, Венера и Земля:



Самостоятельная работа: *Из астрономии мы знаем, что эти планеты имеют различный размер. Давайте сверимся с данными и попробуем сделать размер этих планет более реалистичным. В целом, рекомендуется сделать все планеты чуть меньше, чтобы пользователю было удобнее брать их в руки и переносить.*

Размещение зон

Итак, у нас должна получиться примерно следующая картина:



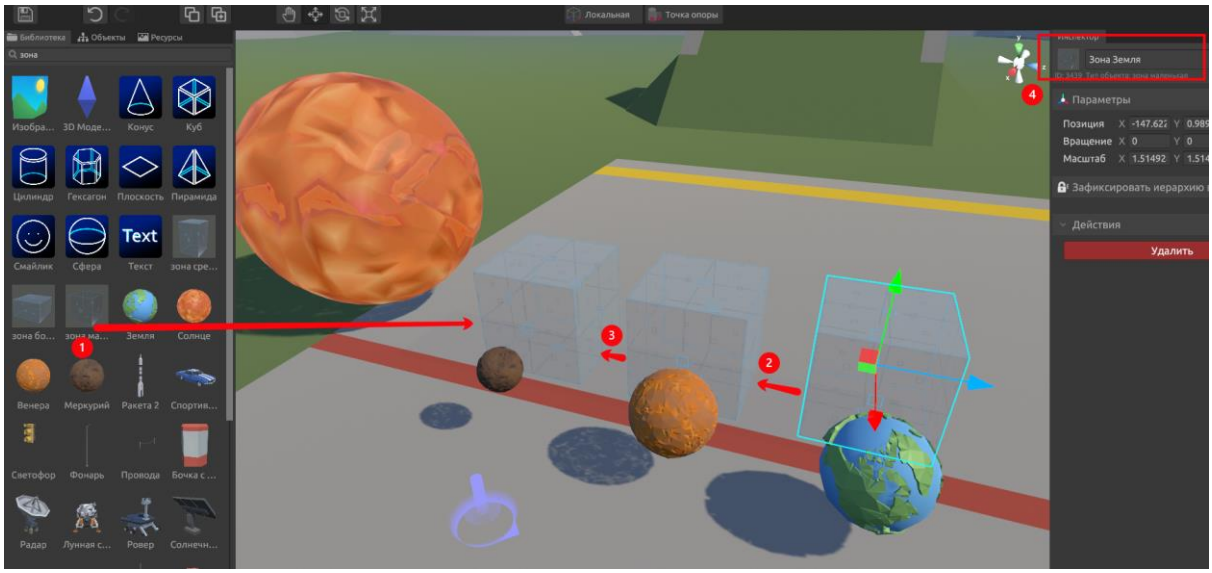
Из ТЗ: *В приложении должна присутствовать механика, позволяющая определить на правильное место мы поставили планету или нет.*

Для того, чтобы реализовать эту механику нам лучше всего подойдет объект “Зона”, который является вспомогательным объектом Varwin.

Определение:
Вспомогательный объект Varwin *- это объект, который выполняет исключительно логическую функцию и не предполагает, что пользователь будет взаимодействовать с ним напрямую. Такой объект виден при размещении на сцене в редакторе логики, но в режиме просмотра приложения он будет невидим для пользователя.*

Разместим три зоны на одной линии, относительно солнца, сделаем их размеры удобными для размещения планет (советуем по размерам ориентироваться на самую большую планету, чтобы она помещалась в зону). Это пункты 1,2,3 на картинке.
Совет*: не забывайте использовать функцию копирования (CTRL+C - копировать, CTRL+V - вставить), чтобы расставлять объекты быстрее.*

Каждой зоне зададим название, для дальнейшего удобного поиска нужной зоны в редакторе логики (4). Например, зона, где должна будет находиться “Земля” дадим имя “Зона Земля”.



Сохраните проект.
Попробуйте запустить проект в режиме Desktop, чтобы убедиться - в режиме просмотра зоны невидимы.

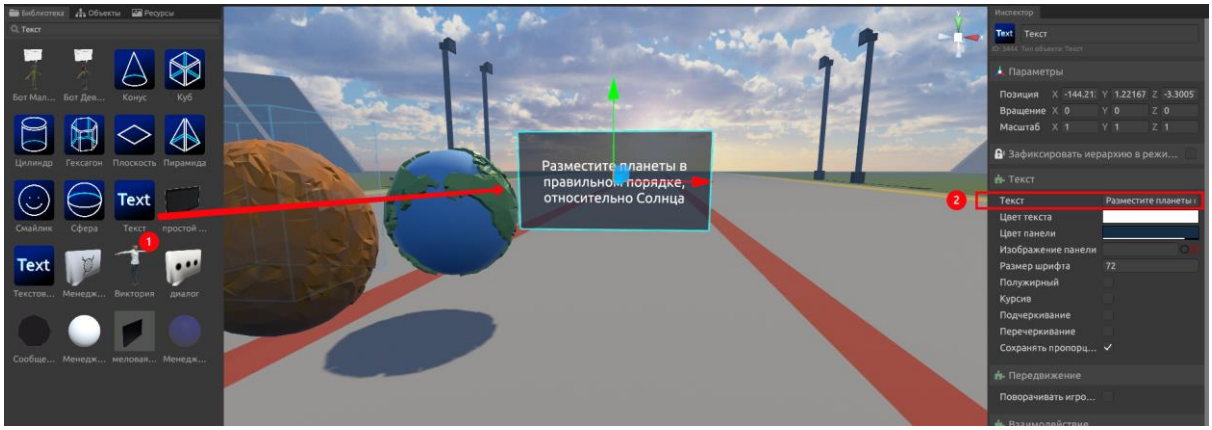
Создание UX/UI-дизайна

Поскольку зоны невидимы в режиме просмотра проекта, то логично, что пользователю будет не до конца ясно, в какие позиции ему будет необходимо разместить планеты, нам необходимо решить эту проблему.
Из ТЗ: *В приложении должен быть реализован удобный UX/UI-дизайн, позволяющий пользователю понять что ему необходимо сделать и в какие точки размещать планеты.*
Определение UX/UI:

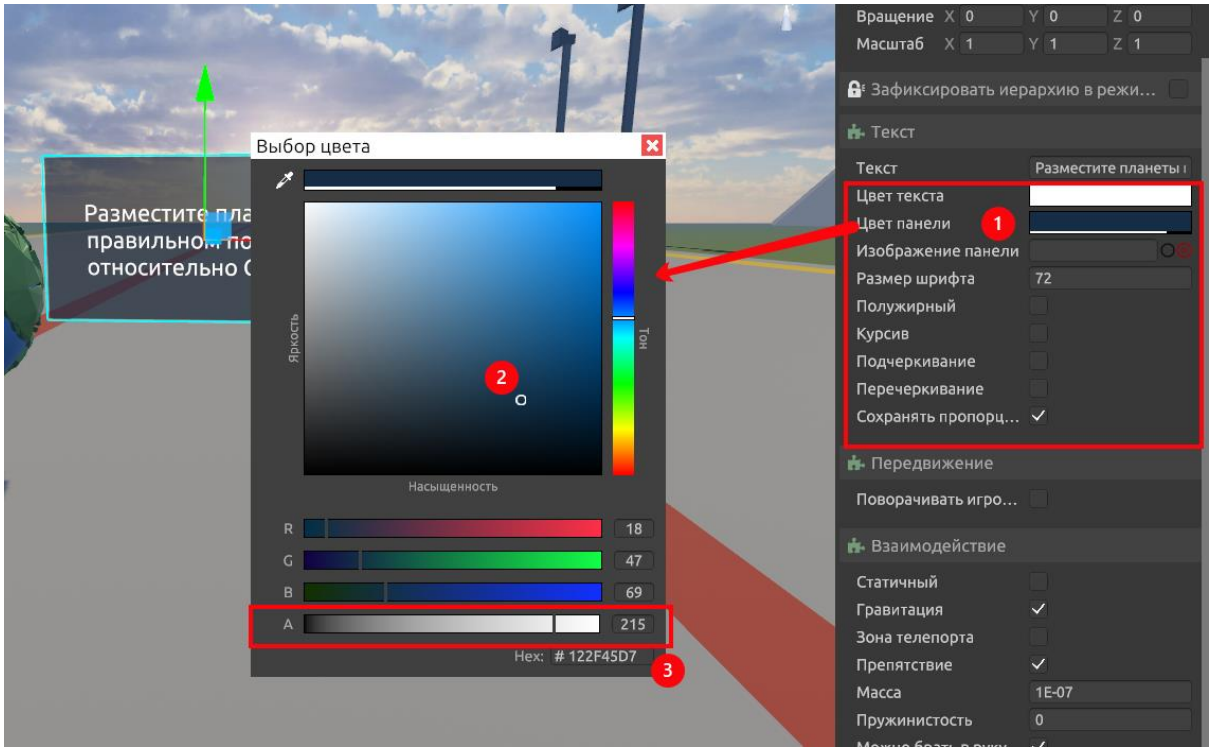
Мы уже знаем, что такое UI – user interface, пользовательский интерфейс - это дизайн кнопок, текстовых панелей, полей ввода – всех точек взаимодействия пользователя с приложением.
UX – user experience, пользовательский опыт - это то, какой опыт/впечатление получает пользователь от работы с вашим интерфейсом. Удается ли ему достичь цели и на сколько просто или сложно это сделать (удобно ли читать текст, взаимодействовать с объектом, насколько вообще ему понятен сценарий вашего приложения).
UX/UI дизайн — это проектирование любых пользовательских интерфейсов в которых удобство использования так же важно, как и внешний вид.

Работа с объектом “Текст”, свойства “Текста

Для того, чтобы пользователю было понятно что делать при запуске приложения добавим UI - объект “Текст” (1).
Выделите этот объект, на правой панели (**инспектор**) вы увидите свойства текста категории “Текст”.
В первую очередь давайте впишем в поле “Текст” формулировку задания для пользователя (2).



Поменяйте цвет панели, для этого нажмите на прямоугольник (1).
Во вспомогательном окошке вы можете выбрать цвет, просто выбрав мышкой нужную область на цветовом поле (2). Или с помощью ползунков “RGB”.
Чтобы изменить прозрачность используйте нижний ползунок “A”.



***Самостоятельная работа**: поиграйте с другими свойствами из категории "Текст", создайте свой уникальный UI.*

***Совет**: старайтесь размещать текст так, чтобы его было удобно читать пользователю, на уровне глаз и достаточного размера. В нашем случае лучше всего разместить текст напротив точки появления игрока, чтобы он сразу увидел текст задания. Чтобы это понять протестируйте приложение в режиме просмотра.*

Базовые свойства типа "Взаимодействие"

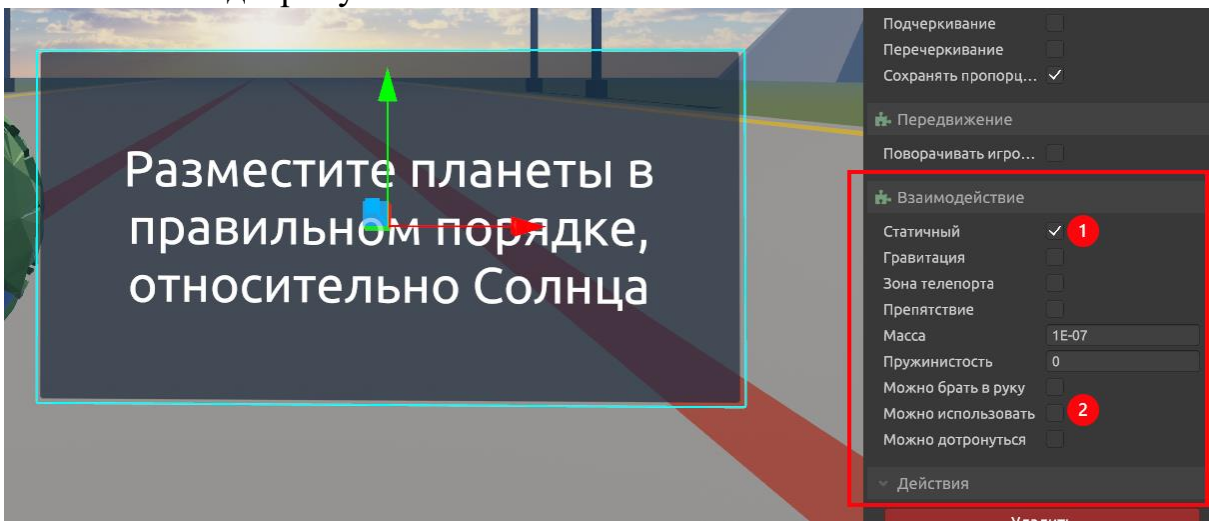
Теперь давайте разберемся с базовыми свойствами объектов из категории *"Взаимодействие"* на примере текста. Пока не вдаваясь в детали, на начальном интуитивном уровне.

Поскольку текст является UI, то предполагается, что он будет просто висеть в воздухе, игрок не должен с ним взаимодействовать (только читать).

1. Для этого нам нужно отключить для объекта гравитацию, чтобы он не упал при запуске приложения.
2. Текст не должен являться препятствием, чтобы игрок и другие объекты могли свободно проходить сквозь него.
3. Текст не является зоной телепорта, мы не можем на него "прыгнуть".
4. Текст является статичным объектом, не предполагается, что он будет перемещаться по сцене и сдвигаться другими объектами. Поставим соответствующие галочки (1).

Чтобы пользователь никак не мог взаимодействовать с текстом контроллерами, то снимаем все галочки (2):

1. Можно брать в руку.
2. Можно использовать.
3. Можно дотронуться.



Это довольно стандартная схема для свойств объекта типа UI, которая пригодится нам в следующем разделе. Запомните ее.

Размещение точек установки планет.

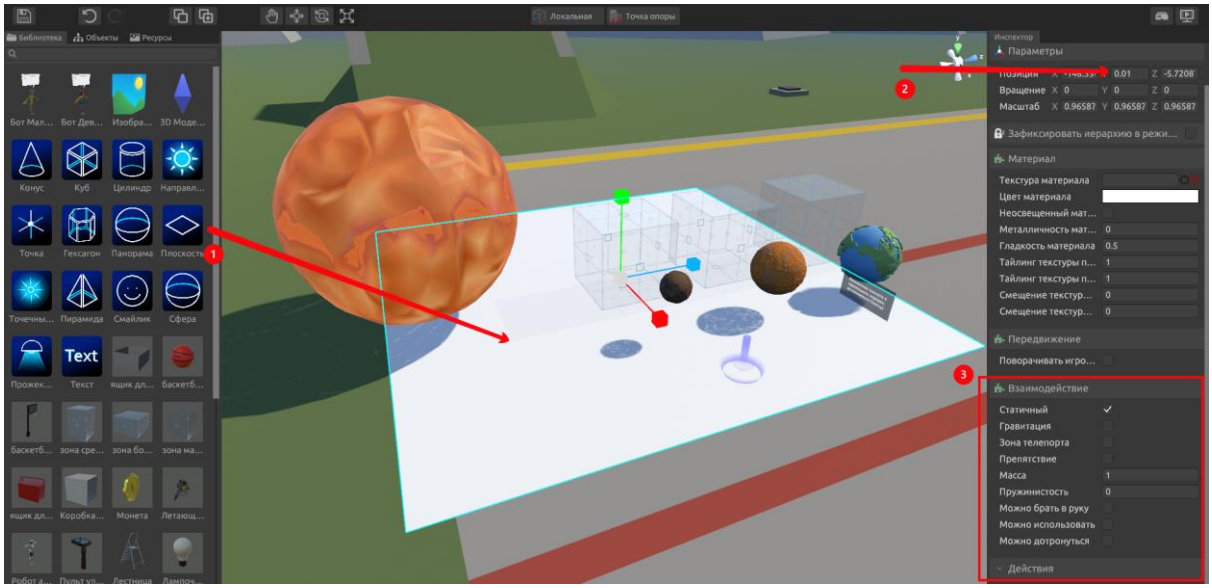
Вторым шагом нам необходимо дать понять пользователю в какие зоны необходимо разместить планеты. Также обратимся к UI.

Для этого используем объект *“Плоскость”* и разместим на сцене (1).

Чуть-чуть поднимем плоскость над землей, чтобы она не сливалась с текстурой “пола” (2).

Можете вручную вписать координату для удобства.

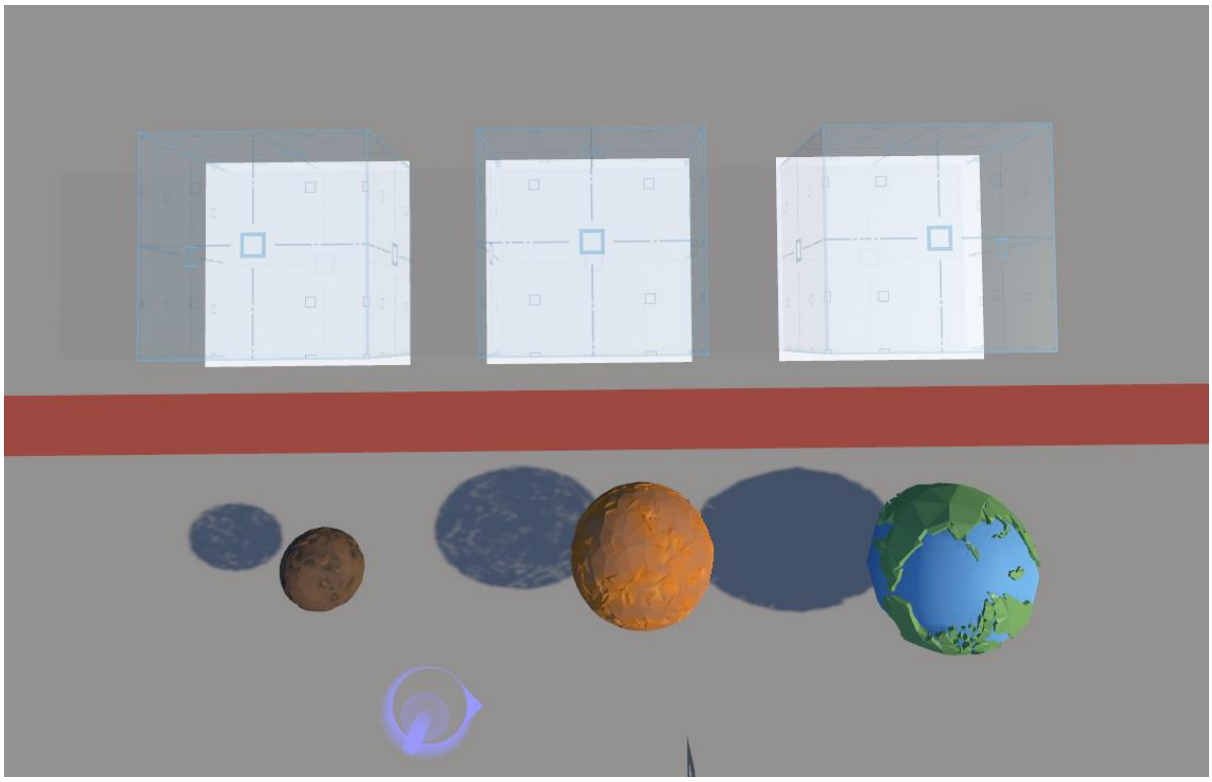
А также применим к плоскости те же самые свойства взаимодействия, что и к тексту (3).



***Примечание**: Обратите внимание, что у плоскости также как и текста можно поменять цвет в свойствах Материала, можете этим*

воспользоваться. Другие свойства Материала мы рассмотрим в следующих темах.

Самостоятельная работа: *вы уже знаете достаточно, чтобы добиться результата как на картинке ниже, сделайте это!*



Размещение объектов для проверки результата

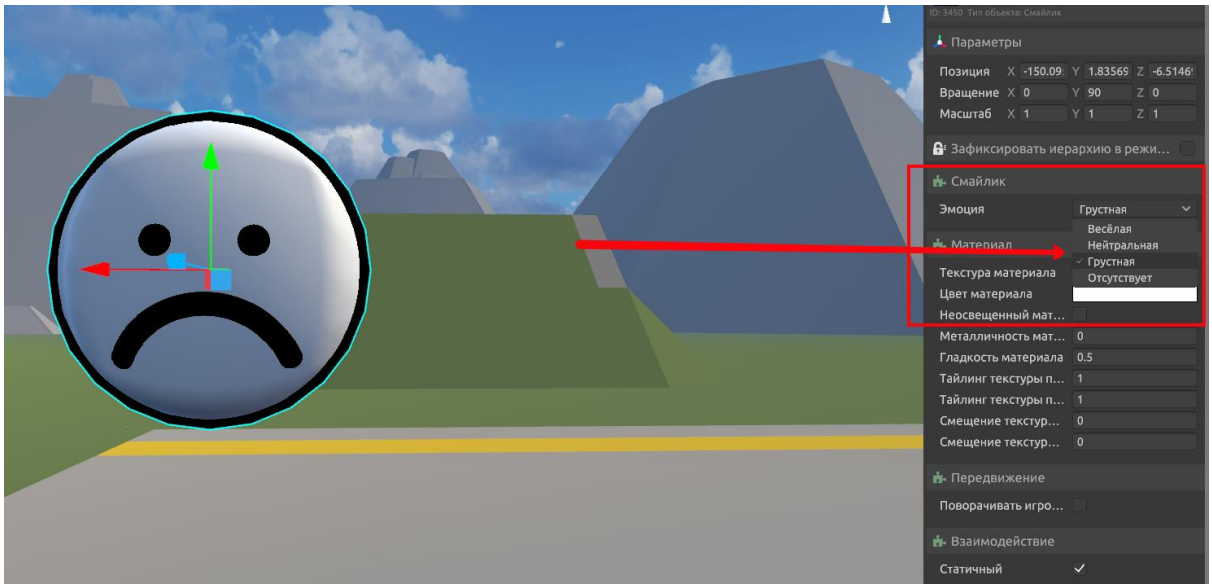
Из ТЗ: *В приложении должна присутствовать механика, позволяющая определить на правильное место мы поставили планету или нет.*

В целях реализации этой механики в данном проекте предлагается использовать объект *“Смайлик”*.

Смайлик в зависимости от условий может менять свою эмоцию: безразличие, грусть, радость.

Давайте разместим эти объекты на сцене (1).
Не забываем давать логичные названия, соответствующие планетам (2).
Поскольку фактически смайлик тоже является UI, то не забываем и о правильных свойствах взаимодействия.

Логика в будущем будет строиться на том, что, если мы помещаем планету в правильную зону, то смайлик становится веселым.
Поэтому давайте сразу зададим грустную эмоцию смайлика через *инспектор*.



Совет: *у некоторых объектов есть свои уникальные свойства, которые можно настраивать через инспектор, как у смайлика. Обращайте на это внимание.*

На следующем занятии мы начнем разрабатывать логику для нашего проекта, поработаем с редактором логики Blockly, реализуем проверки соответствия планет и проверку условия победы.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое техническое задание?
2. Для чего нужен объект “зона” и какой у него тип?
3. Какие продвинутые свойства объектов вы знаете?
4. Какие 4 эмоции есть у объекта “смайлик”?

Занятие 3.3 Настройка логики для зон

Цель:

Модернизировать сцену проекта “Планеты” и применить логические конструкции, согласно техническому заданию на практическую и самостоятельную работу.

Задачи:

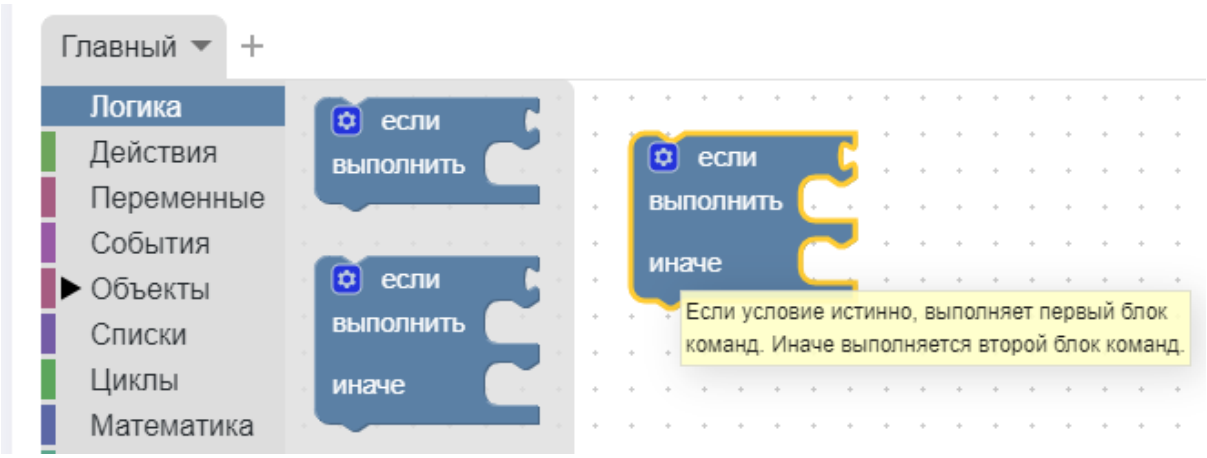
- Познакомиться с объектом “Текст”
- Создать логику для объектов на сцене
- Закрепить понимание условных операторов, логический оператор “И”,
- Продумать и реализовать визуализацию условия победы

Методические материалы для подготовки к занятию:
Пошаговая инструкция по разработке кейсов #3.

Создание логики приложения
Прежде чем открыть редактор логики начать собирать “пазл” из блоков, необходимо повторить определение условного оператора (или оператора ветвления).
Определение: ***Условный оператор*** *- это конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.*

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Таким образом, условный оператор — это средство ветвления вычислительного процесса.

В нашем редакторе логики он выглядит так:



Создание условия

Еще раз вернемся к ТЗ: *В приложении должна присутствовать механика, позволяющая определить на правильное место мы поставили планету или нет.*

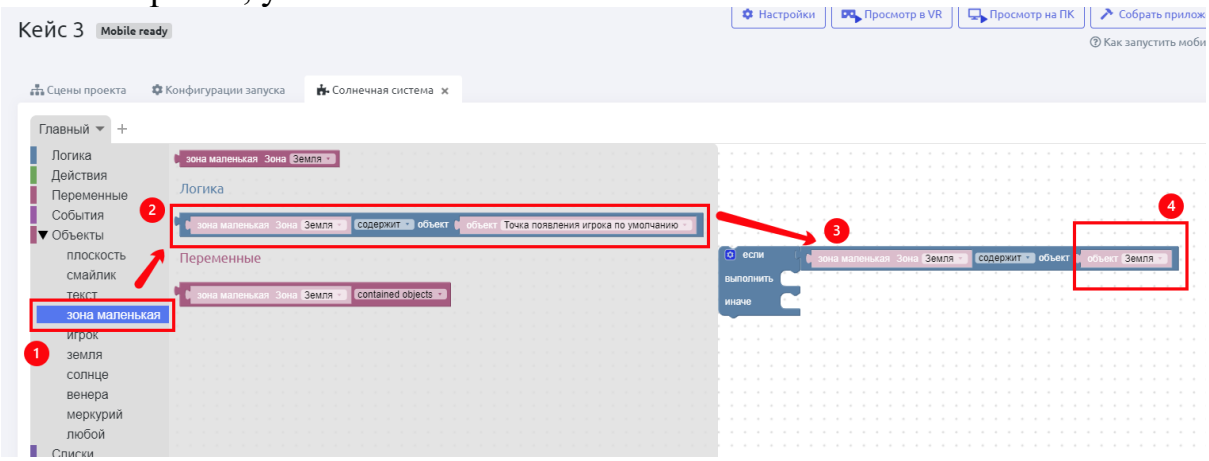
То есть, истинным условием для нас является нахождение определенной планеты в определенной зоне.

Поэтому давайте перейдем в категории объекты и выберем объект *“Зона”* (1).

Вы видите, что зона имеет логический блок *“<Зона X> содержит <объект Y>”* (2), давайте вытащим его и установим напротив *“Если”* (3).

В качестве зоны выберем “Земля” и в качестве объекта тоже *“Земля”* (4). Еще раз вспомним о важности логичных наименований, без этого вы бы не смогли быстро сопоставить нужную зону и нужный объект.

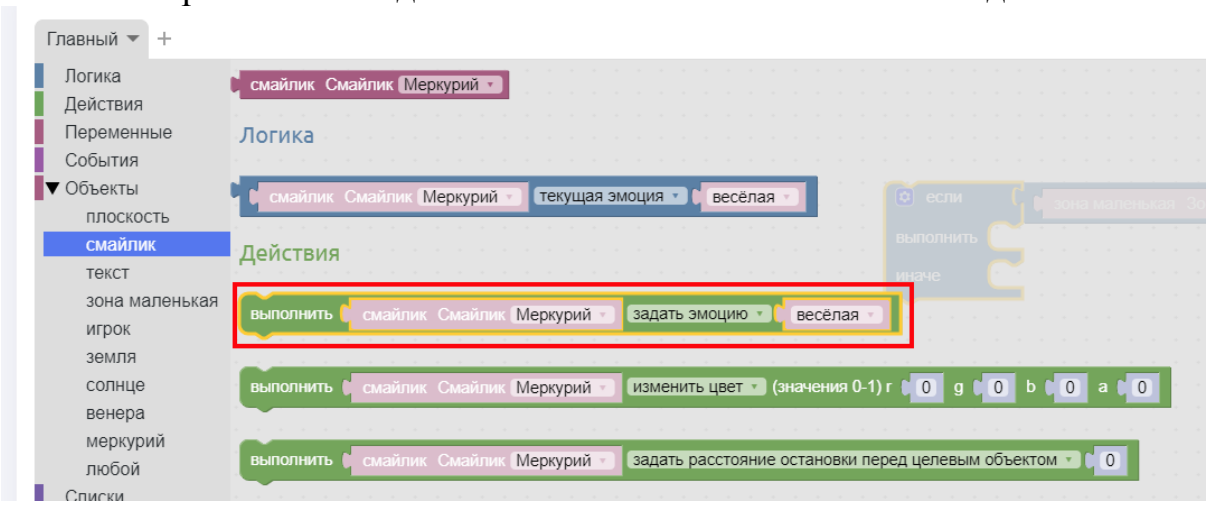
Таким образом, условия готовы!



Создание действий

Мы уже обозначили выше, что при условии нахождении правильной планеты в правильной зоне смайлик меняет эмоцию на веселую.

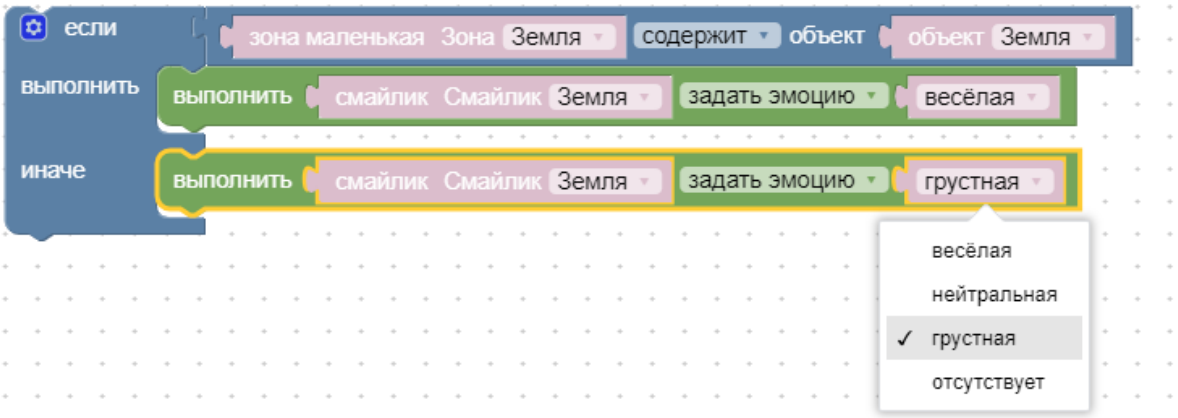
Выберем в блоках действий объекта *“Смайлик”* необходимый блок:



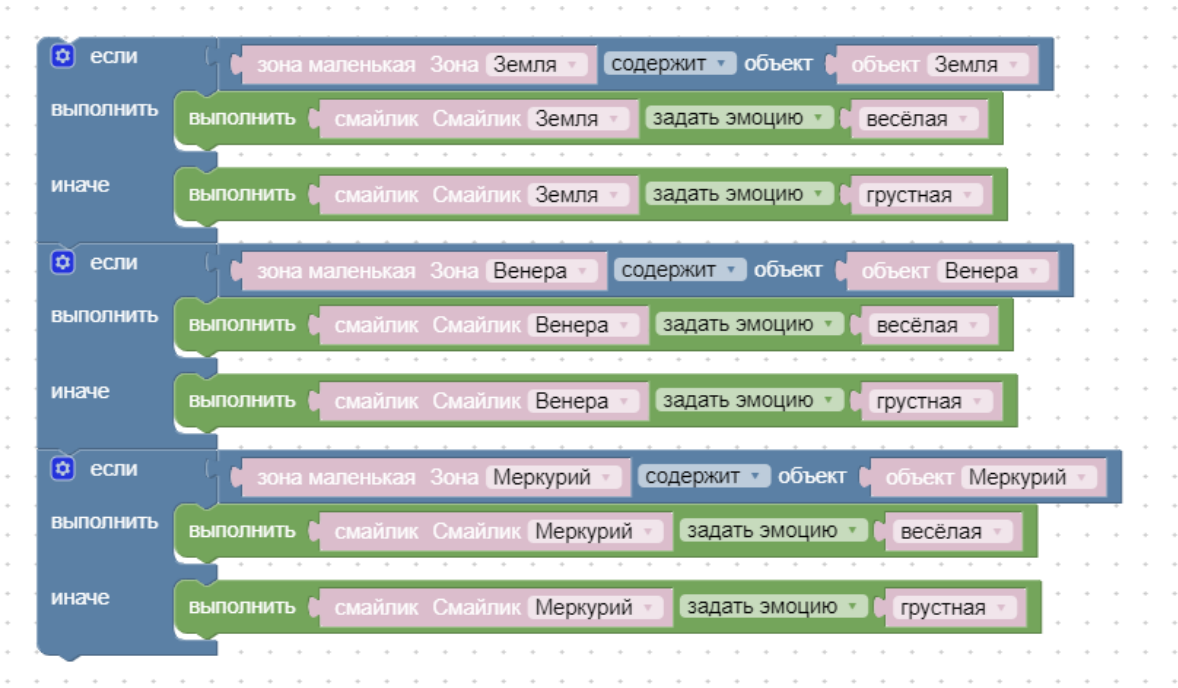
И установим его напротив *“Выполнить”*.

А иначе, соответственно, если планета будет не в своей зоне эмоция будет грустная.

Скопируйте блок действия смайлика и подставьте его напротив “Иначе”, выбрав другую эмоцию. Для *“Земли”* выберите *смайлик “Земля”*, еще раз вспомним про имена :)



Замечательно!
Теперь осталось выполнить то же самое для всех остальных планет, чтобы получилось так:



Это ваша небольшая *Самостоятельная работа.*

Совет: *все это можно сделать за 15 секунд, если вы вспомните как копировать блоки. Вы можете копировать блок "Если" целиком со всеми действиями и условиями, останется только поменять наименования.*
Примечание: *В этом примере не обязательно соединять блоки "Если" между собой, функционально это ни на что не влияет. Такой внешний вид собран для более изящного вида Blockly.*

Условие победы

Мы сделали практически все, осталось выполнить последнее условие из ТЗ: *После размещения всех планет на своих местах должно произойти событие, которое явно бы показывало пользователю о том, что он все сделал правильно.*

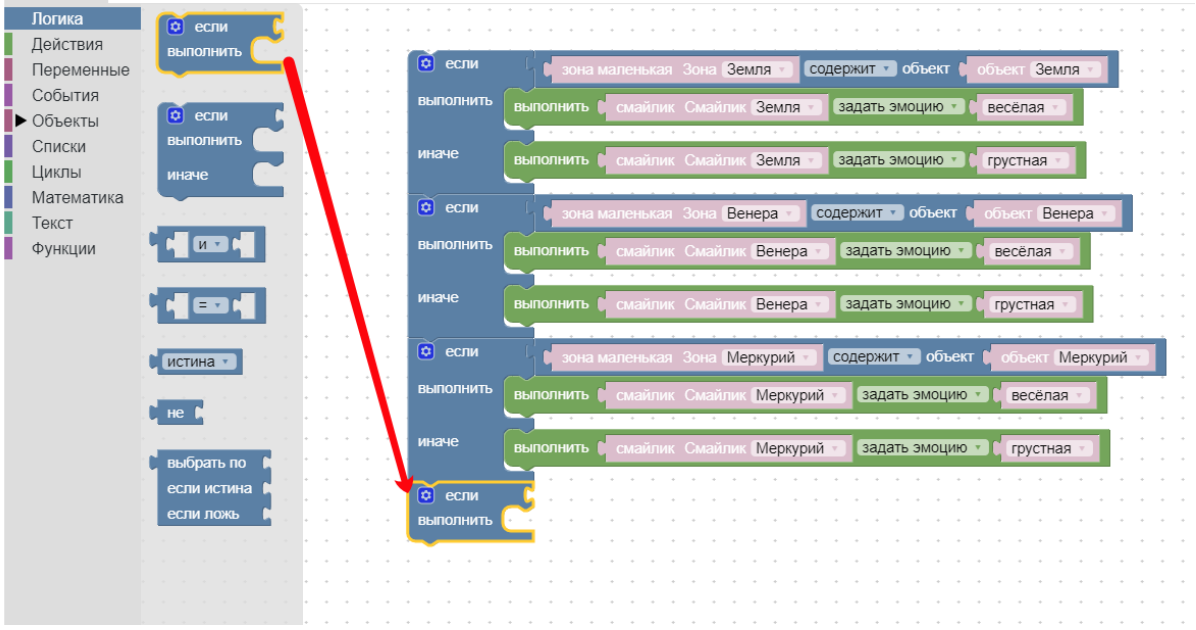
Поскольку у нас уже есть текстовая панель, то предлагается разместить на ней текст *“Все планеты на правильных местах, вы победили!”*, при условии, когда все смайлики будут веселыми.

Для того, чтобы это сделать познакомимся с **логическим оператором “И”**.

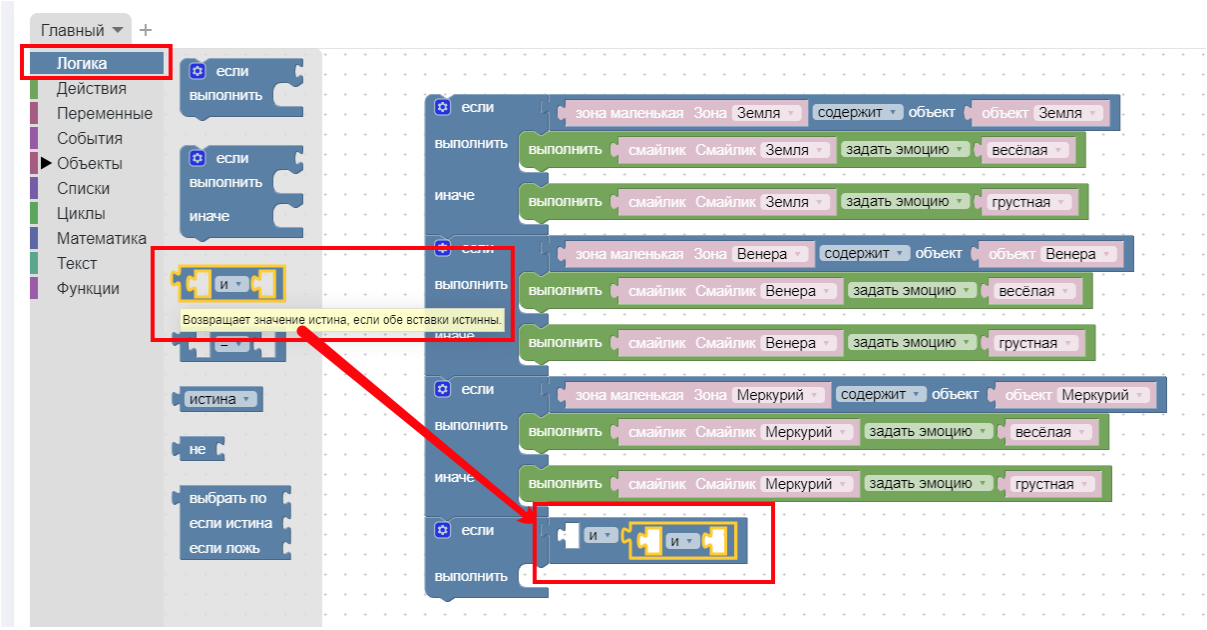
Когда мы проверяем есть планета в зоне или ее нет - это всего лишь одно условие, но бывают ситуации, когда нужно проверить сразу несколько условий, при этом они должны выполняться одновременно.

Здесь нам поможет *логический оператор “И”* - только при условии, что все условия будут выполняться, логический оператор И будет истинным.

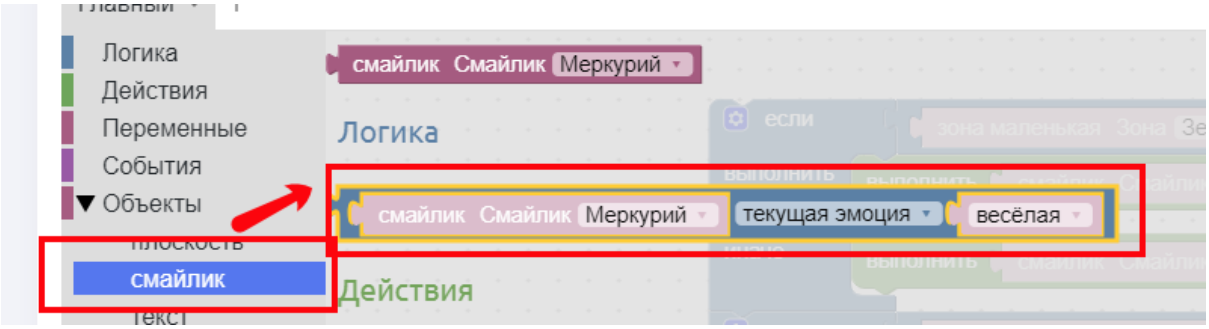
Чтобы понять это на практике добавим еще один условный оператор *“Если”*, на этот раз без *“Иначе”*, поскольку в этом случае это будет лишним.



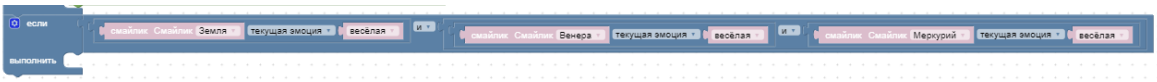
Нам необходимо проверить 3 условия: 3 смайлика должны быть веселыми, поэтому добавляем напротив *“Если”* два *логических оператора “И”* (один вкладываем в другой):



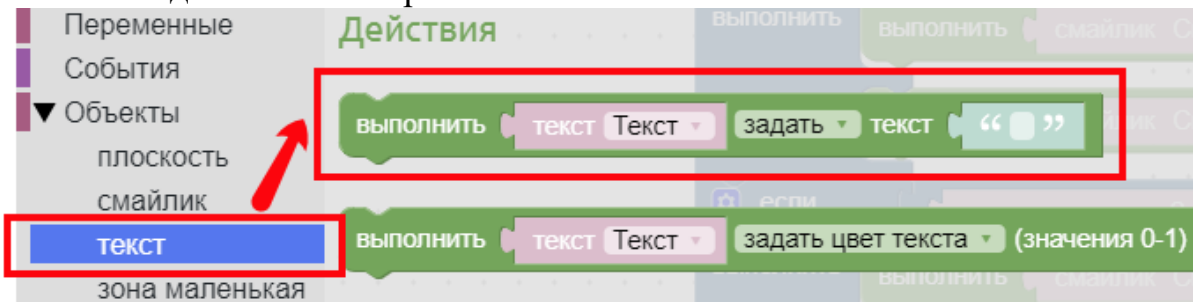
Таким образом у нас получилось 3 пустых ячейки, которые мы заполняем условиями, которые находятся в блоках объекта *“Смайлик”*:



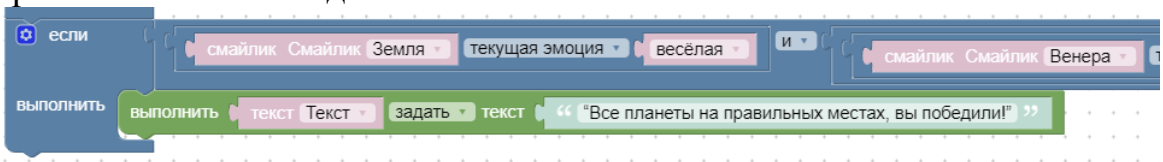
Дословно: если смайлик *“Земля”* веселый И смайлик “Венера веселый И смайлик “Меркурий” веселый, то произойдет какое-то действие.



В качестве действия выбираем блок объекта *“Текст”*:



И прописываем необходимый текст:

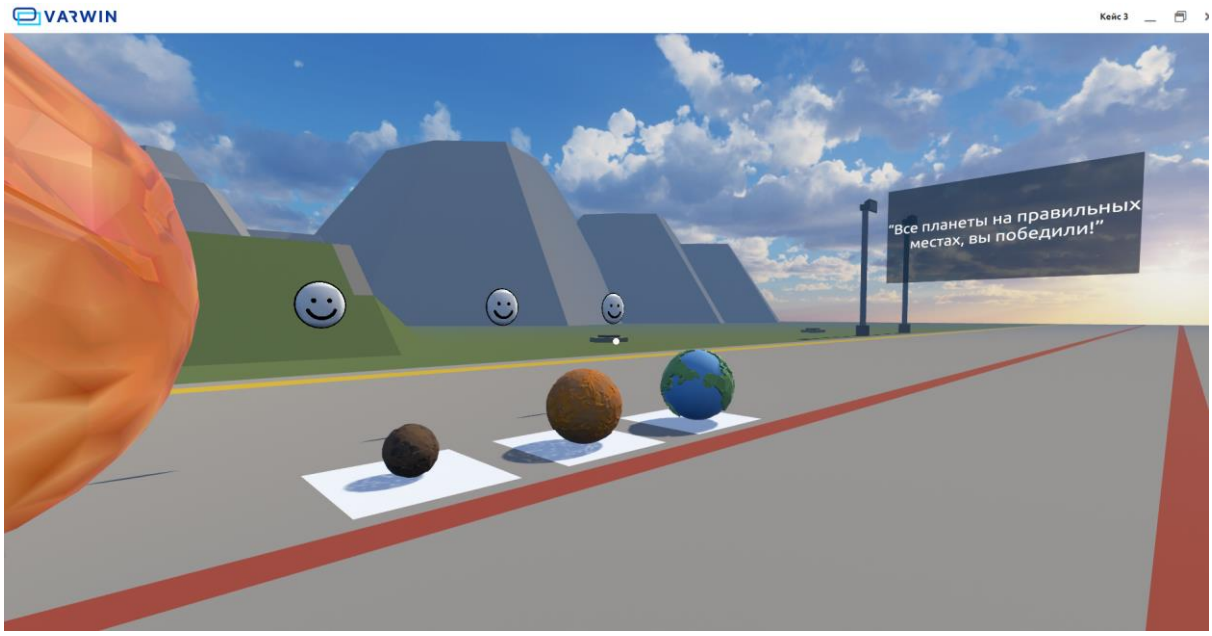


Готово!

Не забывайте нажимать на кнопку **“Применить”** для сохранения логики!

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

Если вы все сделали правильно, то можно запускать проект и проверять. В итоге вы увидите примерно такой кадр:



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что обязательно нужно сделать по итогу расположения блоков в Blockly?
2. Как работают логические операторы “И” и “Или”?
3. Что говорят правила хорошего тона программиста по вопросу именования объектов?

Занятие 3.4 Расширение солнечной системы

Цель:

Усвоение навыков, полученных в ходе практических занятий. Развитие (апгрейд) проекта “Планеты” до проекта “Солнечная система” по собственному техническому заданию.

Задачи:

- Повысить навыки пространственного мышления
- Получить навыки рисования скетчей/ планов перемещения по виртуальному пространству
- Усвоить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Научить обучающихся формировать и фиксировать техническое задание проекта
- Усвоить базовые навыки работы с логикой блоков “Условий” в Blockly
- Закрепить навыки работы с desktop - редактором XRMS Varwin
- Усвоить навыки работы с локацией и размещением объектов на сцене
- Сформировать понимание правильности использования дизайна интерфейсов, основанном на удобстве для конечного пользователя

Формат: Самостоятельная работа, решение кейсового задания.

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #3.

На этом этапе нужно усвоить навык формирования и фиксации технического задания.

Шаблон технического задания:

1. Тематика/направленность проекта
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы)
3. Дополнительные возможности
4. План-схема сцен, которые будут реализованы
5. Каким образом будет реализован UX/UI - дизайн?

Совет: четко и грамотно модерируйте время на формирование собственного технического задания. Обучающиеся могут сильно увлечься и расплывчато формировать ТЗ. Здесь нужно четко и конкретно (желательно тезисами) сформировать ТЗ и нарисовать примерный план сцены (расположенных на ней объектов).

10 - 40 минут	Разработка проекта согласно техническому заданию.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

40 - 45 минут	Сохранение проектов. Тестирование и демонстрация проектов на VR-HMD устройствах.

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Кейс:

Разработать полноценный проект для изучения солнечной системы, используя все планеты из пакета “Астрономия”. Использовать улучшенное событие (действие при соблюдении условий победы) правильной расстановки планет.

Дополнительное задание, если позволяет время: реализовать дополнительные логические конструкции, которые должны быть

зафиксированы в ТЗ. Например, применение сферических 360 панорам в проекте.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта
2. Нарисовать план расположения объектов на сцене
3. Использовать все основные планеты солнечной системы
4. В приложении должна присутствовать механика, позволяющая определить на правильное место мы поставили планету или нет.
5. Реализовать улучшенное событие правильной расстановки планет.

Занятие 3.5 Переменные в Varwin

Цель:

Сформировать понимание переменных в программировании и узнать как они используются в XRMS Varwin

Задачи:

- Понять как и для чего используются переменные
- Узнать какие типы данных существуют и как они применяются в программировании
- Научиться создавать переменные в XRMS Varwin и использовать их в своих проектах
- Понять какие типы переменных используются в XRMS Varwin
- На реальных примерах познакомиться с переменными в XRMS Varwin

Методические материалы для подготовки к занятию:

1. Пошаговая инструкция по разработке кейсов #4.
2. Переменная
3. Переменная в программировании
4. ГЛАВА 3. ПРОГРАММИРОВАНИЕ НА ПАСКАЛЕ :: 3.4. Типы данных :: 24
5. Типы данных и их виды в языках программирования от Loftblog

Определение:

Переме́нная в функциональной и логической парадигмах программирования, переменная определяется как **имя**, с которым может быть *связано* значение, или даже как **место** (location) для хранения значения.

Область видимости и/или время существования переменной в некоторых языках задается классом памяти.

Литература для подготовки к занятию:

Переменная в программировании

Какие типы переменных бывают? Поговорим об основных из них.

Литература для подготовки к занятию:
ГЛАВА 3. ПРОГРАММИРОВАНИЕ НА ПАСКАЛЕ :: 3.4. Типы данных :: 24
Типы данных и их виды в языках программирования от Loftblog

На следующем занятии нам необходимо будет реализовать один из пунктов ТЗ:

Должна быть создана система оценки, позволяющая определить количество правильных ответов баллах.

Это значит, что нам необходимо будет не просто зафиксировать, что результат успешный/неуспешный как в предыдущем проекте, а динамически определить результат в баллах.

Для этого удобнее всего использовать переменную, которая будет хранить результат.

Давайте попробуем это сделать!

Из определения переменной мы можем сделать вывод, что у переменной должно быть имя и что ее содержание может изменяться в зависимости от условий.

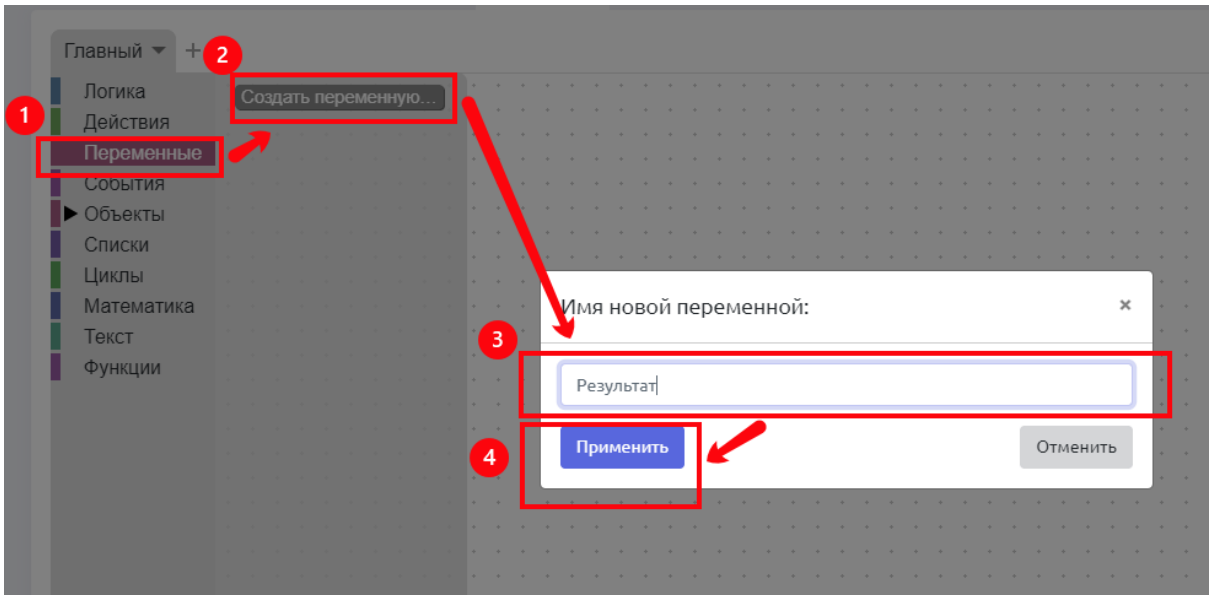
Давайте для начала научимся создавать переменную в Varwin.

Для этого в редакторе логики выберем категорию “Переменные” (1).

Нажимаем на “Создать переменную” (2).

Дадим переменной имя, в данном случае предлагается имя “Результат” (3).

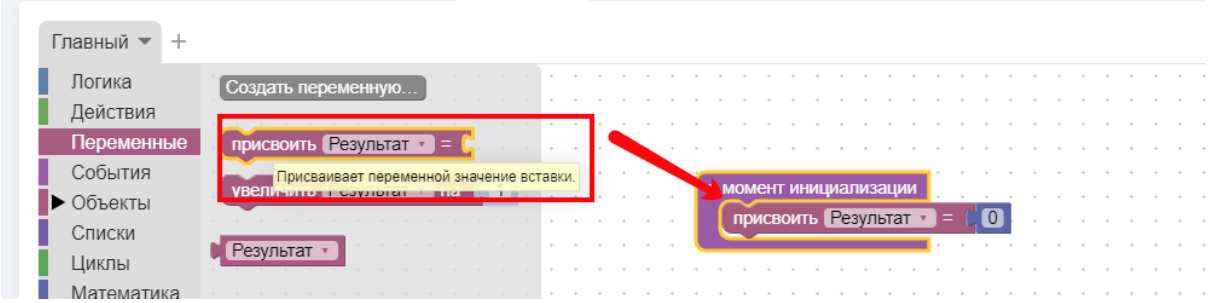
Для завершения создания нажимаем на кнопку “Применить” (4).



Теперь вам необходимо задать начальное значение переменной.

Пока не вдаваясь в детали, выберем категорию “События” и выберем блок “В момент инициализации”. В этот блок подставим конструкцию “Присвоить результат” + в категории “Математика” выберем маленький блок с числом “0”.

Должна получиться следующая конструкция:

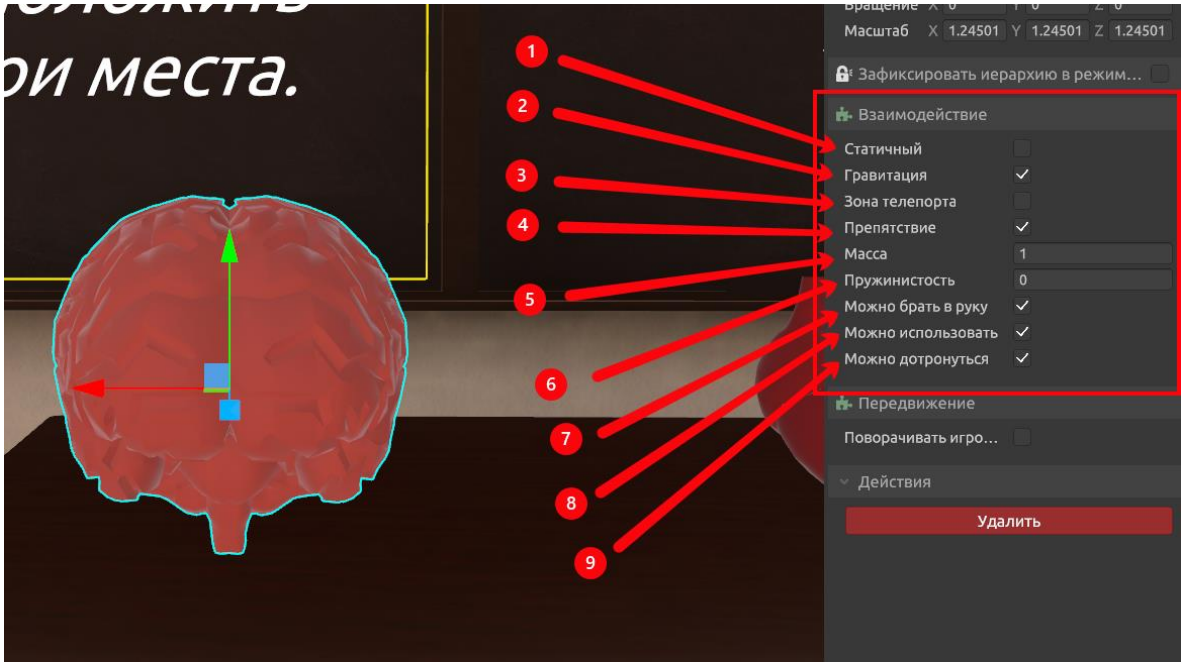


Итак, вы создали переменную и присвоили ей начальное значение в момент инициализации (запуска) приложения. Теперь осталось сформировать условия, при которых эта переменная будет изменяться. Что мы с Вами и сделаем на следующем практическом занятии.

А сейчас давайте подробно разберем все основные свойства взаимодействий, которые присутствуют практически в каждом объекте из библиотеки.

Примечание: у отдельных объектов могут быть уникальные свойства, как например у “Текста”, поэтому обращайте внимания на свойства в инспекторе каждый раз, когда выделяете объект.

Давайте выделим один из объектов на локации, например “Мозг”:



Выделив абсолютно любой объект из библиотеки Varwin Education вы увидите примерно такую картину по свойствам, как на скриншоте выше. Пойдем по порядку:

1. **Свойство “Статичный”** - свойство, которое в целом отвечает за то, будут ли действовать законы физики на объект при взаимодействии с другими объектами, т.е. будет ли сдвигаться объект при столкновении с другими (не статичный) или эти объекты будут проходить насквозь (статичный объект).
2. **Свойство “Гравитация”** - определяет будет ли действовать сила тяжести на объект, упадет ли он на землю при запуске мира или будет висеть в невесомости.
   Примечание: *на статичные объекты гравитация не действует, даже если она для них включена.*
3. **Свойство “Зона телепорта”** - определяет сможет ли игрок переместиться на поверхность объекта с помощью механики телепортации в VR, сможет ли игрок ходить по объекту на клавишах в desktop-режиме просмотра.
   Примечание: *более всего актуально для больших плоских объектов. Например объект “Плоскость” всегда можно превратить в “пол” для перемещения, но об этом в следующем кейсе.*
4. **Свойство “Препятствие”** - определяет будет ли объект являться препятствием для игрока или других объектов или сквозь объект будет проходить абсолютно все.
   Примечание: *объект будет проходить в том числе и через границы локации, поэтому не включайте для объекта без свойства “Препятствие” гравитацию, иначе он просто провалится под пол. Объект-препятствие может быть одновременно статичным, чтобы являться препятствием для игрока, но не для других предметов, это*

может быть полезным для создания UX/UI-дизайна, например, для создания "невидимых стен".

5. **Свойство "Масса"** - определяет физическую характеристику объекта, насколько его будет сложно сдвинуть другим объектом при определенной скорости и т.д. В целом это вполне справедливо и для реального мира.
6. **Свойство "Пружинность"** - насколько далеко объект будет "отпрыгивать от поверхности" при ударе с определенной скоростью. Баскетбольный мяч имеет высокую пружинность, а слиток золота близкую к нулю.
7. **Свойство "Можно брать в руку"** - игрок сможет взять в руку этот объект с помощью кнопки "grab" на контроллере.
8. **Свойство "Можно использовать"** - игрок сможет использовать этот объект по кнопке "use" на контроллере.
9. **Свойство "Можно дотронуться"** - если это свойство активировано, то возможно использовать события, если игрок дотронулся контроллером до объекта (о таких типах событий мы поговорим чуть позже). В момент, когда мы дотрагиваемся до объекта на нем появляется синяя обводка.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое переменная?
2. Для чего используются переменные?
3. Какие типы данных вы знаете?
4. Какие типы переменных используются в XRMS Varwin?

Занятие 3.6 Строение тела человека

Цель:

Разработать основной функционал проекта "Строение тела человека" и применить логические конструкции основанные на использовании переменных.

Задачи:

- Закрепить навык создания переменных в XRMS Varwin и использования их в своих проектах
- Научиться выводить переменные в объект "Текст"
- Научиться задавать события для объектов типа "Зона"
- Научиться присваивать имена и значения переменным
- Усвоить навыки тестирования своих проектов на баги/ошибки
- Получить навыки дизайн-мышления для успешного проектирования UX/UI
- Усвоить навыки использования свойств объекта
- Научиться использовать блоки категории "Событие" в своих проектах

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #4.

В этот раз сразу начнем с ТЗ:

Необходимо разработать приложение на платформе Varwin, которое позволяло бы проверять знания о расположении органов человеческого тела.

Специальные требования:

1. В приложении должны использоваться минимум 3 органа человеческого организма.
2. Должна быть создана система оценки, позволяющая определить количество правильных ответов в баллах.
3. Должен быть реализован удобный UX/UI.
4. Только когда все органы будут на месте, то должно высветиться сообщение о победе.

Подготовительная работа

Уже по классической схеме:

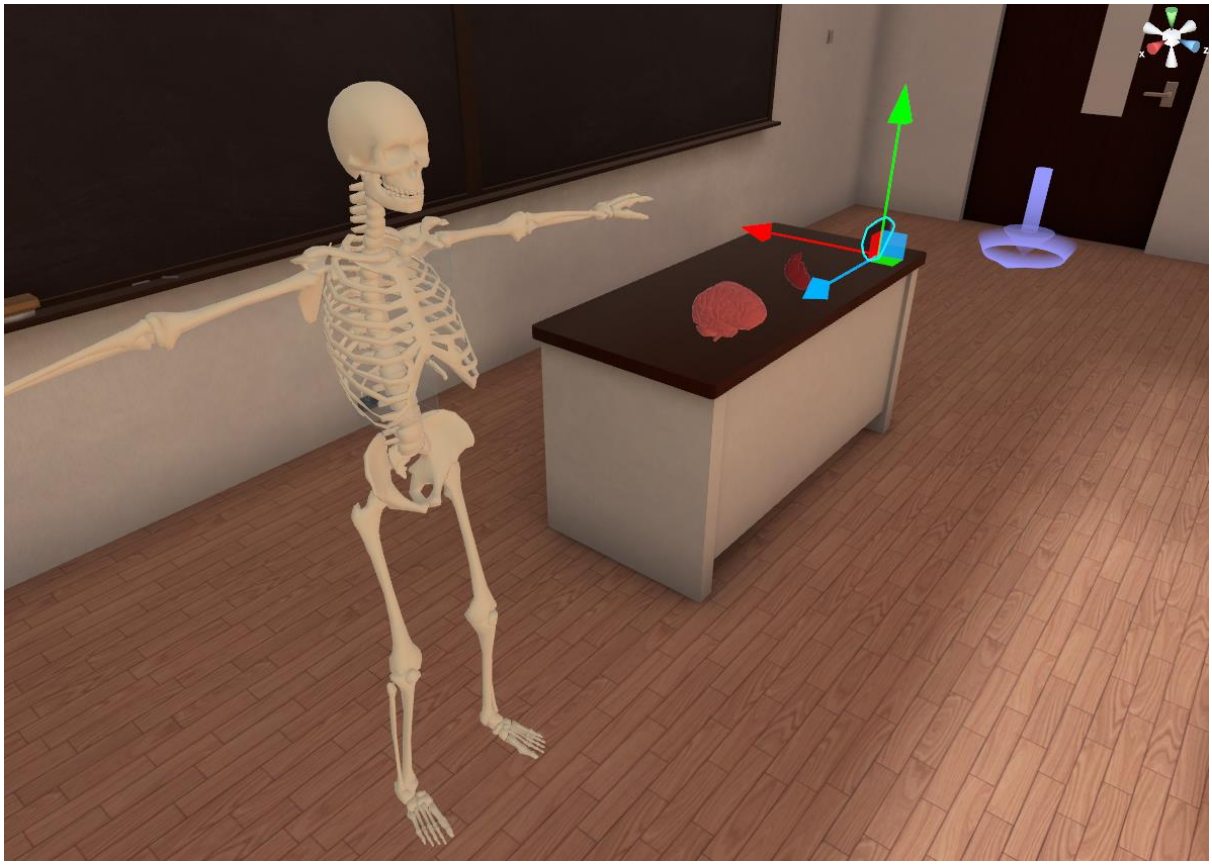
1. Создаем новый проект
2. Выбираем локацию, в данном случае отлично подойдет “Школьный класс”, но вы можете выбрать и другую локацию.
3. Открываем Desktop-редактор.

***Читаем ТЗ**: Необходимо разработать приложение на платформе Varwin, которое позволяло бы проверять знания о расположении органов человеческого тела.*

Специальные требования:

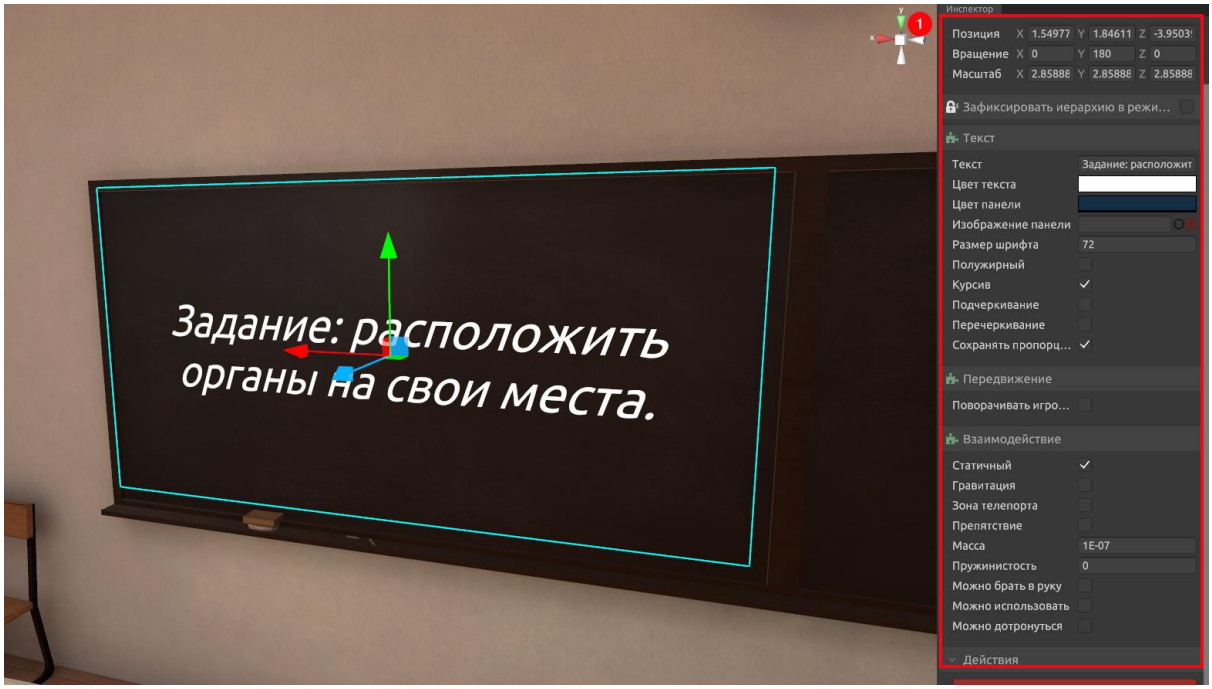
1. *В приложении должны использоваться минимум 3 органа человеческого организма.*

Значит на сцене нам понадобится человеческое тело и 3 органа из библиотеки “Анатомия”: в нашем примере мы выбрали *Скелет, Мозг, Сердце и Печень*.

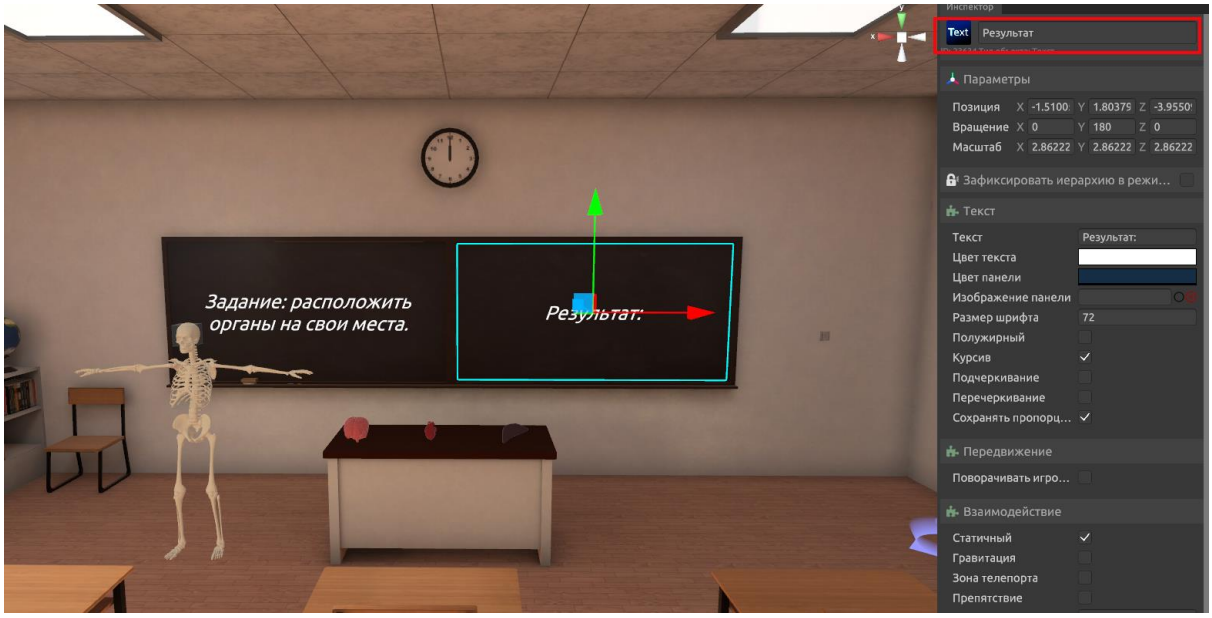


Также нам понадобится ”Текст”, а вернее даже два таких объекта.
На одном мы напишем формулировку задания, а на другой будем отображать результат.

Самостоятельная работа: *Вы уже знакомы с параметрами и свойствами объекта, поэтому постарайтесь добиться такого результата, как будто задание написано мелом на доске (подсказка на картинке ниже).*

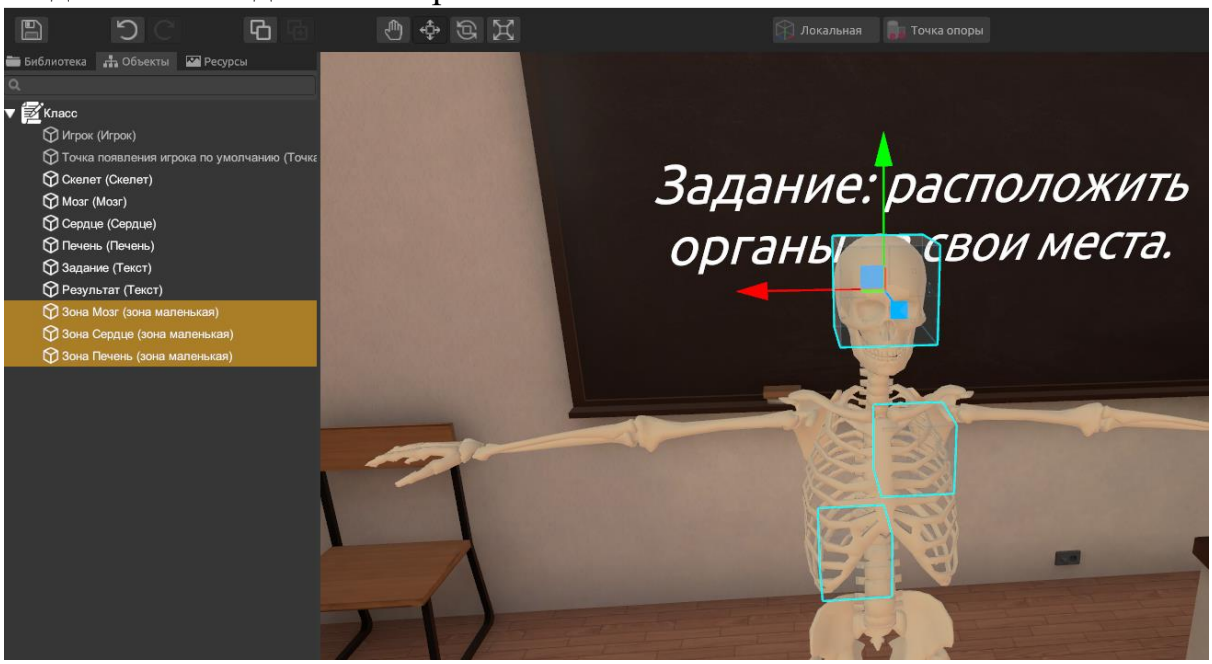


Для того, чтобы сделать вторую панель, просто копируем первую и даем ей название “Результат”, а также поменяем в ней текст:



Самостоятельная работа: *Также нам необходимо разместить 4 зоны в местах, где должны располагаться органы и назвать их соответствующим образом. Где должны располагаться органы вы можете найти в учебнике по анатомии, а как располагать зоны мы уже знаем из предыдущего кейсового задания.*

Подсказка находится на картинке ниже:



Итак, все необходимые объекты на сцене, переходим к настройке их свойств!

Стандартные свойства объектов и их настройка

Давайте подумаем какие свойства применить для объектов на сцене. Для этого применим технику UX/UI-дизайн мышления: продумываем возможные действия пользователя с этими объектами.

Во первых для объекта “Скелет”:

По сути, с точки зрения UX/UI-дизайна у пользователя не должна быть возможность перемещать этот объект, т.к. это может в будущем сломать логику размещения органов (скелет не будет перемещаться вместе с зонами).

Поэтому следует отключить все свойства, связанные с взаимодействием контроллерами, а именно: “Можно брать в руку”, “Можно использовать”, “Можно дотронуться”.

Скелет должен быть статичным объектом, т.к. нам важно, чтобы органы проходили сквозь него, поэтому ставим галочку, напротив “Статичный”.

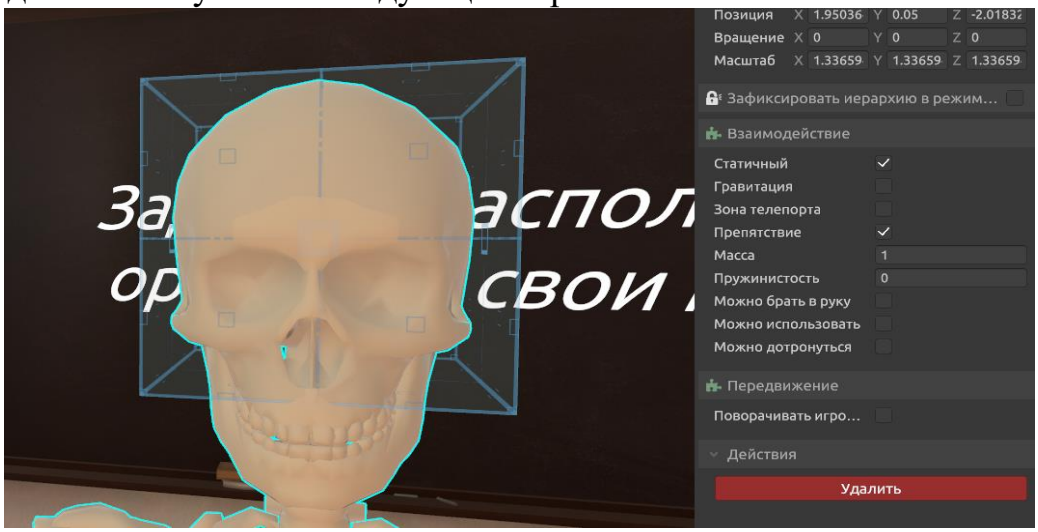
С другой стороны, для пользователя скелет будет являться препятствием, чтобы он не мог сквозь него пройти, поэтому ставим галочку напротив свойства “Препятствие”.

Т.к. скелет уже имеет свойство “Статичный”, то не имеет смысла добавлять свойство “Гравитация”.

“Зона телепорта” нам точно не нужна, поскольку мы не собираемся “ходить” по скелету, это и с точки зрения реалистичности было бы неправильно.

Свойства “Масса” и “Пружинность” здесь не играют особой роли, т.к. скелет не задействован во взаимодействии с другими объектами, поэтому оставим их значения по умолчанию.

В итоге должна получиться следующая картина:



Далее давайте в такой же логике продумаем свойства для органов:

Пользователь должен взаимодействовать с этими предметами, поэтому выставляем все галочки, связанные с контроллерами (забегая вперед, свойства “Можно использовать” и “Можно дотронуться” по факту не будут никак влиять на логику в сценарии, но с другой стороны и не помешают его реализации, поэтому давайте их оставим).

Поскольку органы будут располагаться близко (например сердце и легкие), то нам важно, чтобы они не выталкивали друг-друга из скелета и

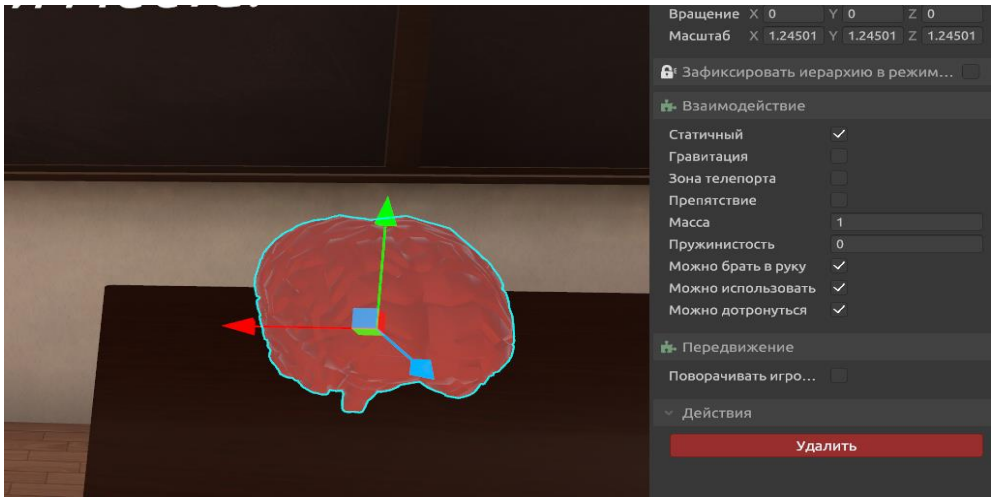
могли проходить насквозь и скелета, и других органов, поэтому выставляем галочку “Статичный”.

“Гравитация” нам так же не понадобится, т.к. органы должны “висеть” в скелете, иначе они упадут. Но в любом случае, поскольку объект “Статичный”, то гравитация не будет работать. Тем не менее уберем галочку со свойства “Гравитация” “для галочки”, чтобы не путаться в будущем.

“Зона телепорта” также нам не понадобится, логика здесь та же, что и для скелета.

Свойство “Препятствие” можно отключить, т.к. органы маленькие, и не будет ничего страшного, если пользователь сможет проходить сквозь них, возможно, в некоторых моментах, это будет даже удобнее.

“Масса” и “Пружинность” здесь также не важна, мы попробуем использовать эти свойства в следующих кейсах.



Самостоятельная работа: *в третьем кейсовом задании мы уже выставляли свойства для UI-объектов, типа “Текст”. В текущем проекте также есть такие объекты. Продумайте почему мы выставили именно такие свойства, применяя дизайн-мышление, а также примите их в текущем проекте.*

Сборка логики

Теперь откроем редактор логики и снова обратимся **к ТЗ:**

Должна быть создана система оценки, позволяющая определить количество правильных ответов баллах.

Это значит, что нам необходимо будет не просто зафиксировать, что результат успешный/неуспешный как в предыдущем кейсе, а динамически определить результат в баллах.

Допустим, если мы поставили только один орган на свое место, а остальные перепутали, то результат будет “1 из 4”.

Для этого удобнее всего использовать переменную, которая будет хранить результат.

Ранее мы с Вами уже создавали переменную, для использования её в данном проекте, давайте сделаем это ещё раз.

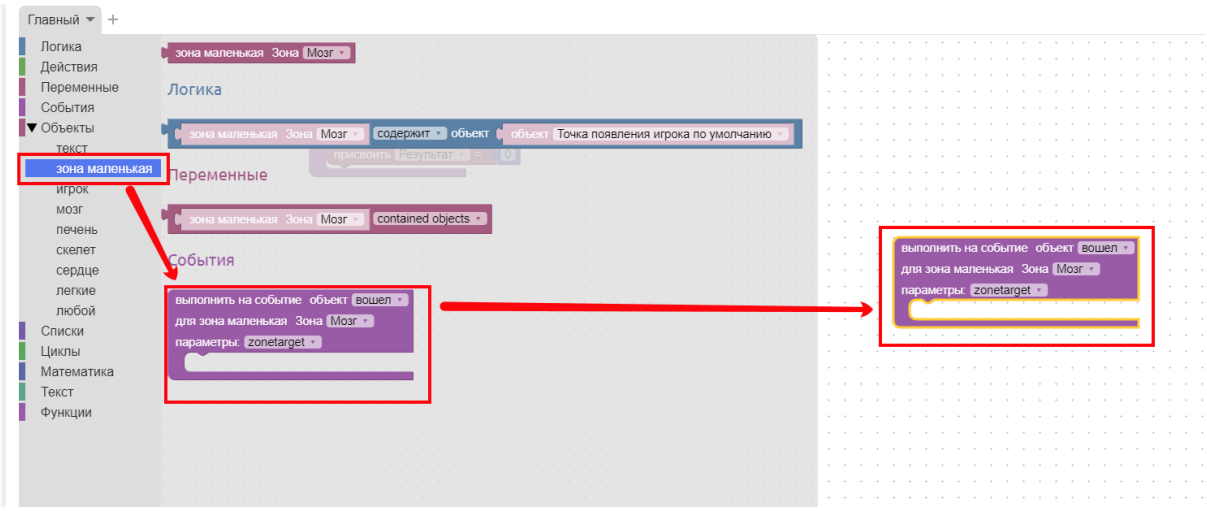
“Итак, вы создали переменную и присвоили ей начальное значение в момент инициализации (запуска) приложения. Теперь осталось сформировать условия, при которых эта переменная будет изменяться.”

Для этого обратимся к новому типу блоков объектов из категории “События” на примере все тех же зон.

Создание логики для зон с помощью Событий

Определение: События *- это блок, который выполняет действие* ***один раз*** *при наступлении определенных условий, прописанных в архитектуре события.*

Для зоны данный блок выглядит так:



Архитектура события состоит из следующих пунктов:

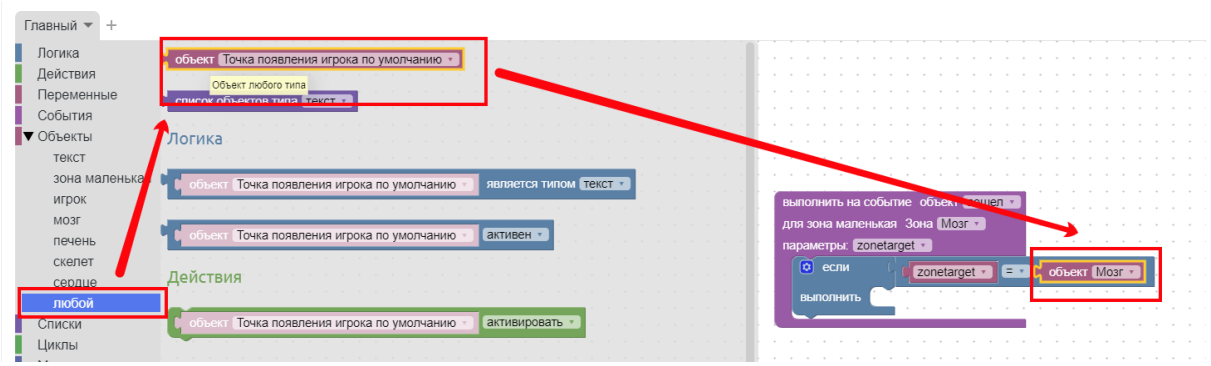
1. Условие выполняется при совершении действия: объект вошел/вышел.
2. Выбор зоны в которую должен войти (или из которой должен выйти объект), задается условным оператором.
3. Параметр “zonetarget” - это переменная, которая определяется объектом. Например для зоны Мозг zonetarget = объект Мозг. Т.е. это объект, который должен войти в/выйти из зоны для активации события.

Давайте соберем первое событие для зоны “Мозг”.

Для этого выберем для пунктов события:

1. Событие: Вошел
2. Зона: Мозг
3. В тело события вставим блок, который будет проверять условие равна ли переменная zonetarget значению объекта “Мозг. Для этого выберем в категории “Объекты” “Любой” и используем самый верхний блок, выбрав из выпадающего списка объект “Мозг”.

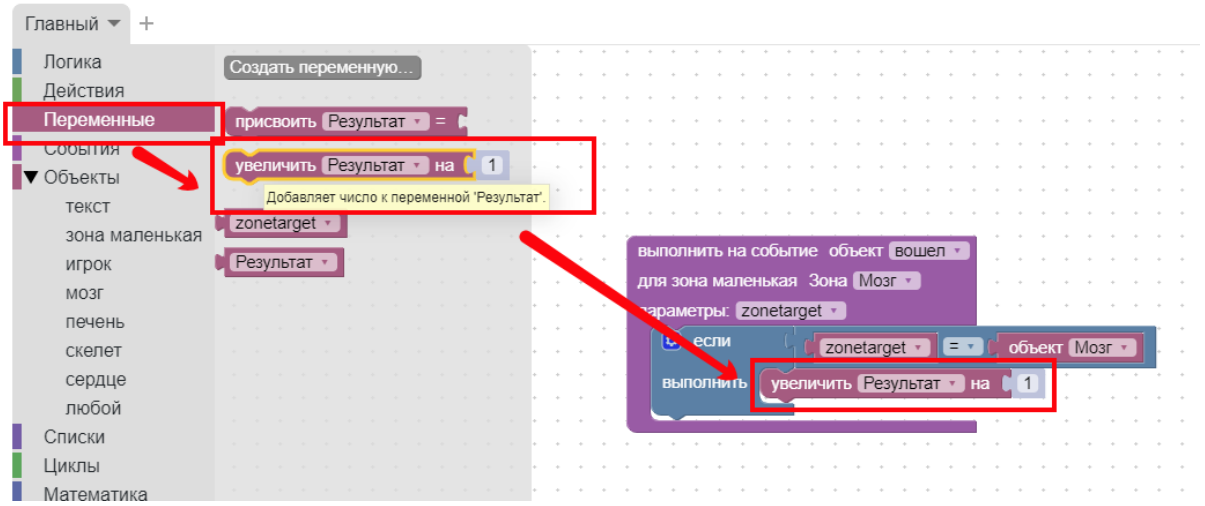
Примечание: “Любые” объекты содержит блоки, которые можно применить абсолютно к любым объектам на сцене, в. т.ч. и к Игроку. Об этом мы поговорим подробнее в следующих кейсах.



После этого необходимо в теле условного оператора определить действия, которые произойдут при его активации.

В данном случае, если мы помещаем в “Зону Мозг” объект “Мозг”, то мы все делаем верно, поэтому переменная “Результат” должна увеличиться на единицу.

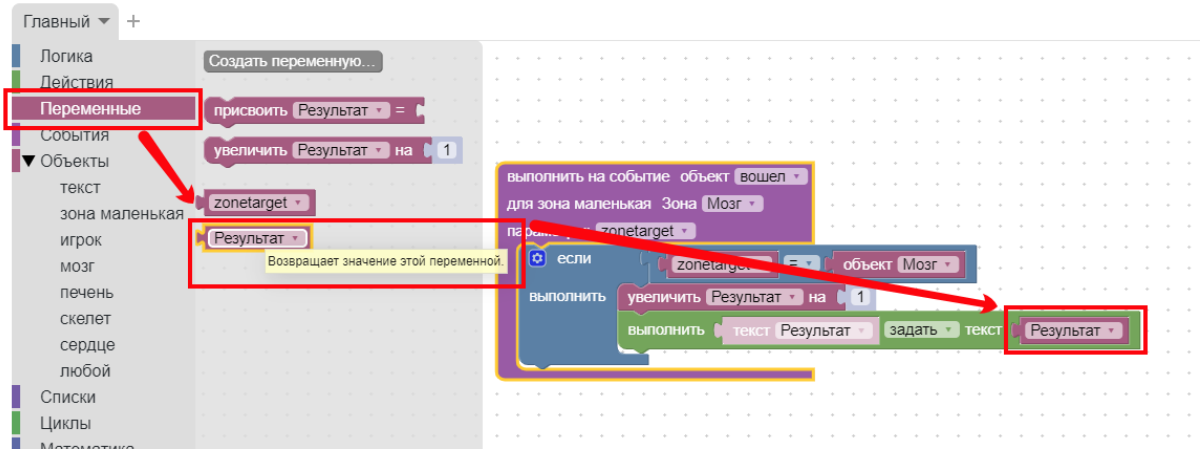
Это очень частая задача в программировании логики приложения, поэтому она вынесена в отдельный блок:



Примечание: *в других языках подобное действие также называют Инкремент, инкрементирование (от англ. increment «увеличение») — операция во многих языках программирования, увеличивающая переменную. Обратную операцию называют декремент (уменьшение). Чаще всего унарная операция приводит переменную к следующему элементу базового типа (то есть для целых чисел — увеличивает на единицу.*

Последнее, что нам следует сделать при активации события - это обновить текстовую панель с результатом.

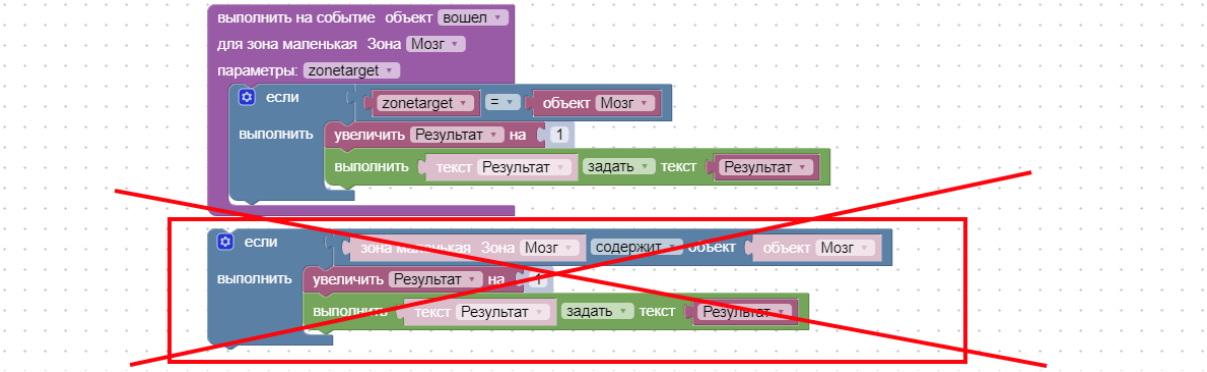
Для этого вставьте блок “Задать текст” в объекте “Текст” в тело события и в поле “Текст” вставьте переменную “Результат”.



Таким образом, каждый раз когда событие будет активироваться, у нас будет обновляться результат на доске.

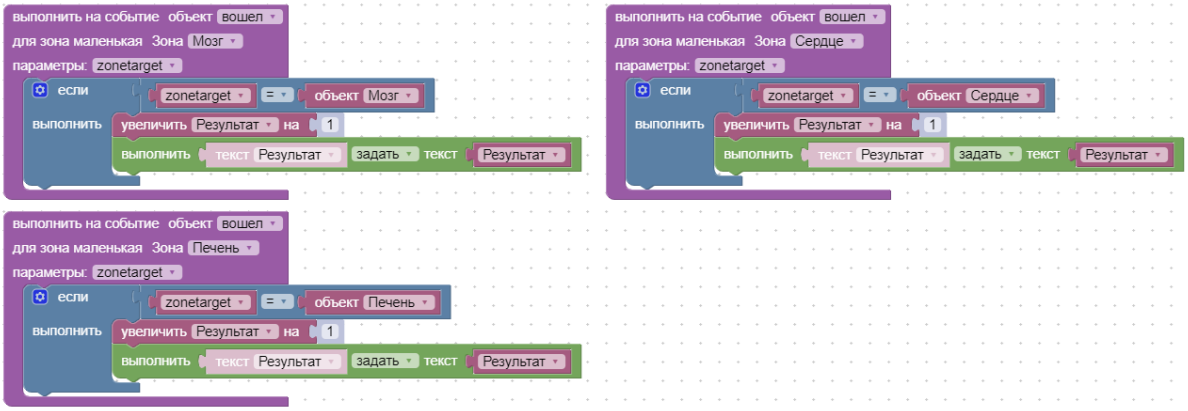
Примечание 1: *Вы уже могли сделать вывод о том, что переменные могут хранить разные типы данных: числовые, текстовые, объекты и др. Разные типы данных мы разберем в следующих кейсовых заданиях.*

Примечание 2: *особенность блоков из категории* ***"Событие"*** *заключается в том, что проверка происходит* ***единоразово*** *при наступлении соблюдения условия. Блок с условным оператором* ***"Если"*** *не подошел бы для реализации задуманной логики, поскольку он постоянно проверяет выполнено ли условие или нет, и если выполнено, то выполняет действия в теле этой логической конструкции. Т.е., если бы мы собрали конструкцию наподобие этой:*



То результат бы бесконечно увеличивался на 1, пока объект "Мозг" находится в "Зоне Мозг". В ситуации же событием, это происходит лишь один раз.

Самостоятельная работа: *теперь соберите такие же события для всех остальных органов, чтобы получилась следующая картинка:*



Создание финального сообщения

Из ТЗ: *Когда все органы будут на месте, то должно высветиться сообщение о победе.*

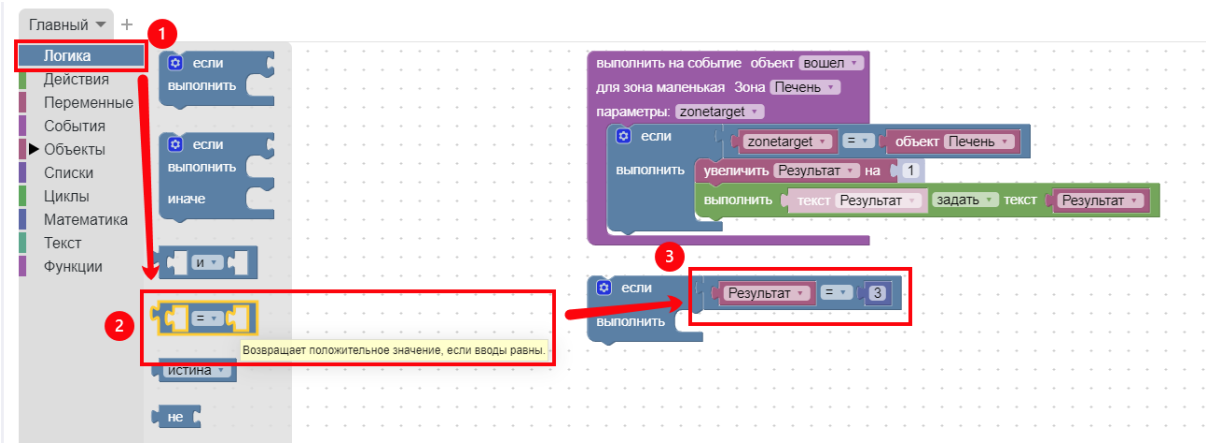
Когда все органы будут на месте, значение переменной будет = 3. Это наше условие для вывода сообщения.

Для этого обратимся к новому блоку из категории “Логика”, который отвечает за сравнение (1).

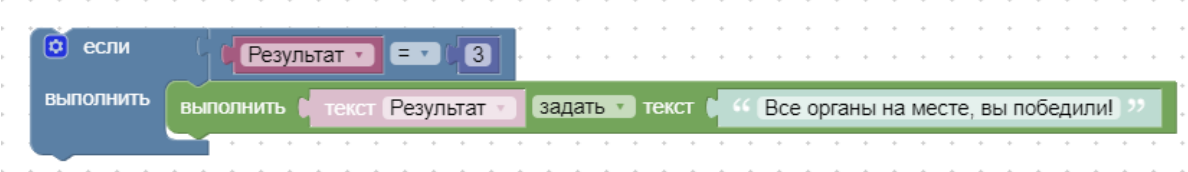
Выберем блок “Сравнение” (2).

Примечание*: этот блок можно развернуть и увидеть другие знаки сравнения.*

Подставим этот блок напротив “Если” и добавим в пустые ячейки “Результат” = “3” (3). Напомним, что это блок из категории “Математика”.



Далее добавьте действие при выполнении условия, вы уже знаете как это сделать:



Тестирование проекта

После того как вы применили логику, можно попробовать запустить проект.

Нужно понимать, что в этом проекте вы разработали более сложный проект, чем в предыдущих, поэтому повышается шанс возникновения ошибок и недочетов в сценарии - это нормально для процесса разработки, но крайне неверно, когда в конечном продукте присутствуют ошибки. Именно поэтому во время разработки любого проекта присутствует стадия тестирования.

Определение: Тестирование *- это процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.*

В данном учебном курсе мы не будем глубоко погружаться в методику тестирования (или QA (от англ. quality assurance) — обеспечение качества), самое главное для Вас - это запомнить, что любой проект перед его финальной презентацией необходимо проверять несколько раз на наличие ошибок.

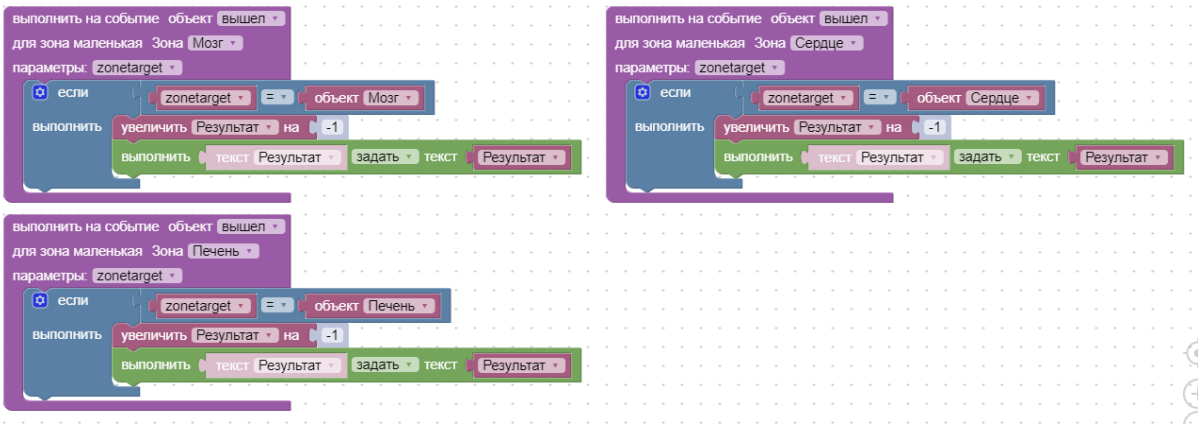
Запустите проект и попробуйте ответить на вопрос: *что в проекте работает неправильно? Зафиксируйте эти замечания.*

Наверняка вы найдете самый грубый недочет - это то, что переменная у нас увеличивается на 1 каждый раз, когда объект попадает в свою зону.

Это значит, что просто поместив и вынув “Мозг” 3 раза подряд можно победить, а что еще хуже, после победы, повторяя эти действия, переменная снова начнет увеличиваться на доске, что ну совсем не удовлетворяет нашему изначальному **ТЗ:**

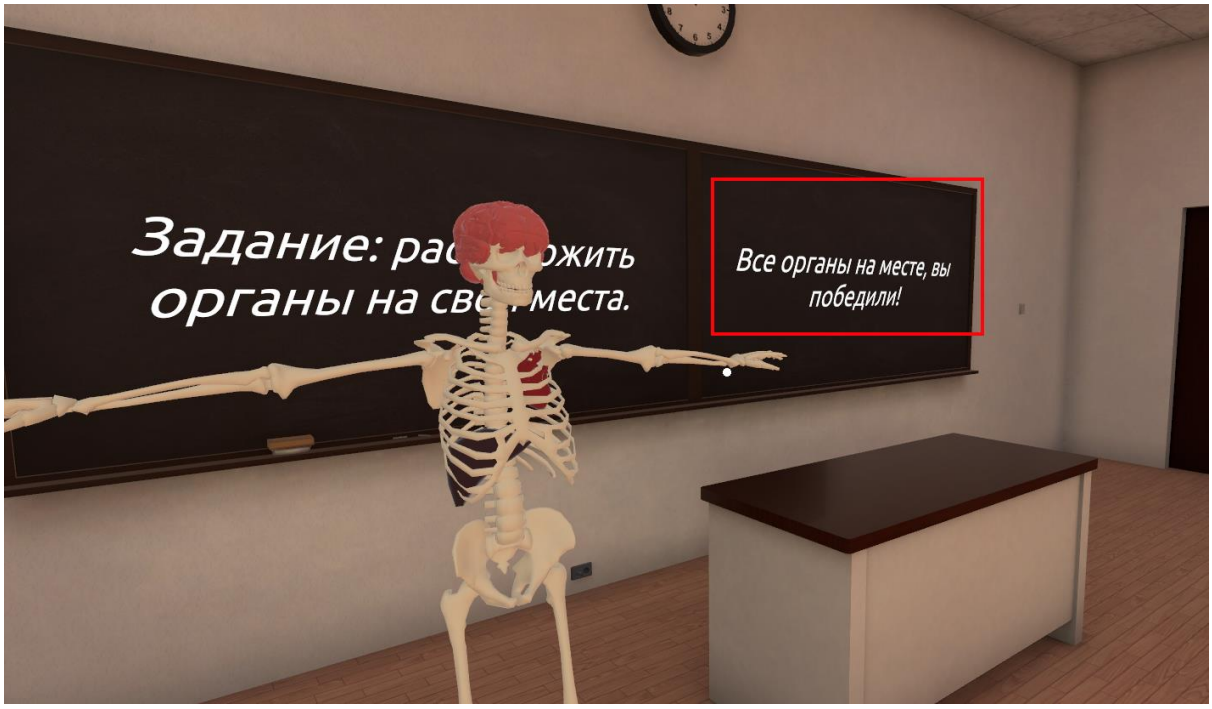
Только когда все органы будут на месте, то должно высветиться сообщение о победе.

Давайте решим этот недочет с помощью дополнительных блоков, которые будут отнимать на 1 значение переменной при выходе целевого объекта из своей зоны:



Примечание: когда мы отнимаем от результата единицу - это называется “Декремент”, обратная операция по отношению к “Инкременту”.

Если вы все сделали правильно, то при размещении всех органов вы увидите следующую картину:



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое тестирование проекта? Что под собой подразумевает этот термин?
2. Как можно было по-другому реализовать систему оценки правильности расположения объектов/органов на скелете?
3. Что такое “zonetarget” и как он работает?
4. Сколько раз нужно проводить тестирование проекта?
5. Что такое инкремент и декремент?
6. Что будет если включить свойство “гравитация” для объекта без включенного свойства “препятствие”?

Занятие 3.7 Расширение проекта (Строение тела)

Цель:

Усвоение навыков, полученных в ходе практических занятий. Развитие (апгрейд) проекта “Строение тела человека” до проекта “Квиз по анатомии” по собственному техническому заданию.

Задачи:

- Повысить навыки пространственного мышления
- Повысить навыки рисования скетчей/ планов перемещения по виртуальному пространству
- Закрепить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Усвоить базовые навыки работы с логикой блоков “Условий” в Blockly
- Закрепить навыки работы с desktop - редактором XRMS Varwin
- Усвоить навыки работы с локацией и размещением объектов на сцене

- Сформировать понимание правильности использования дизайна интерфейсов, основанном на удобстве для конечного пользователя

Формат: Самостоятельная работа, решение кейсового задания.
Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.
На этом этапе нужно усвоить навык формирования и фиксации технического задания.

Литература для подготовки к занятию:

Техни́ческое зада́ние (ТЗ, техзада́ние) — документ, содержащий требования заказчика к объекту закупки, определяющие условия и порядок ее проведения для обеспечения государственных или муниципальных нужд, в соответствии с которым осуществляются поставка товара, выполнение работ, оказание услуг и их приемка.

Шаблон технического задания:

1. Тематика/направленность проекта
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы)
3. Дополнительные возможности
4. План-схема сцен, которые будут реализованы
5. Каким образом будет реализован UX/UI - дизайн?
6. Какую полезную нагрузку несет проект, или будет нести при дальнейшей доработке?

Совет: четко и грамотно модерируйте время на формирование собственного технического задания. Обучающиеся могут сильно увлечься и расплывчато формировать ТЗ. Здесь нужно четко и конкретно (желательно тезисами) сформировать ТЗ и нарисовать примерный план сцены (расположенных на ней объектов).

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Кейс: Квиз по анатомии

Необходимо разработать проект, позволяющий изучать строение внутренних органов человека в формате игры-опроса. С возможностью усвоения базовых знаний в области анатомии.

Дополнительное задание, если позволяет время: реализовать дополнительные логические конструкции, которые должны быть зафиксированы в ТЗ. Например, вывод переменной на отдельную текстовую панель, использование булевых переменных в проекте, сообщение о размещении объекта в неправильной зоне.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта

2. Нарисовать план расположения объектов на сцене
3. Зафиксировать дополнительные функции, которые будут реализованы в проекте.
4. В приложении обязательно должна присутствовать механика опроса для получения знаний по анатомии.

4 Стандартная логика и примитивы

Занятие 4.1 Типы примитивов в Varwin

Цель:

Познакомиться с понятием “примитив” в трёхмерной графике и узнать какие свойства есть у примитивов в XRMS Varwin.

Задачи:

- Узнать что такое примитивы
- Познакомиться со свойствами примитивов
- Научиться настраивать свойства примитивов в XRMS Varwin
- Понять как использовать ресурсы для примитивов
- Научиться взаимодействовать с примитивами в Desktop-редакторе Varwin

Методические материалы для подготовки к занятию:

На этом этапе нужно дать понять что такое трехмерная графика и с чего она начинается, откуда появилась.

Литература для подготовки к занятию:

Трёхмерная графика

Рассказать что такое примитивные объекты и для чего они используются. Откуда всё начинается, уточнить про двумерные примитивные объекты и как их можно использовать в трехмерном моделировании.

Литература для подготовки к занятию:

Геометрический примитив - Geometric primitive - qwe.wiki

Графические примитивы

Трехмерные примитивы

Трехмерные примитивы составляют основу многих программных пакетов компьютерной графики и обеспечивают возможность создания разнообразных объектов простой формы. Часто для формирования нужной модели трехмерные примитивы приходится объединять или модифицировать. Есть два типа примитивов.

Стандартные (Standard Primitives):

Box (Параллелепипед), Sphere (Шар), Geosphere (Геосфера), Cone (Конус), Cylinder (Цилиндр), Tube (Труба), Torus (Тор), Pyramid (Пирамида), Teapot (Чайник), Prism (Призма).

Дополнительные (Extended Primitives:

Hedra (Многогранник), Torus Knot (Тороидальный узел), Chamfer Box (Параллелепипед с фаской), Chamfer Cylinder (Цилиндр с фаской), Oil Tank (Цистерна), Capsule (Капсула), Spindle (Веретено) и другие.

Определение:

Примитивы *в 3D - дизайне - это простые геометрические объекты, с помощью которых можно реализовать базовое наполнение локации.*

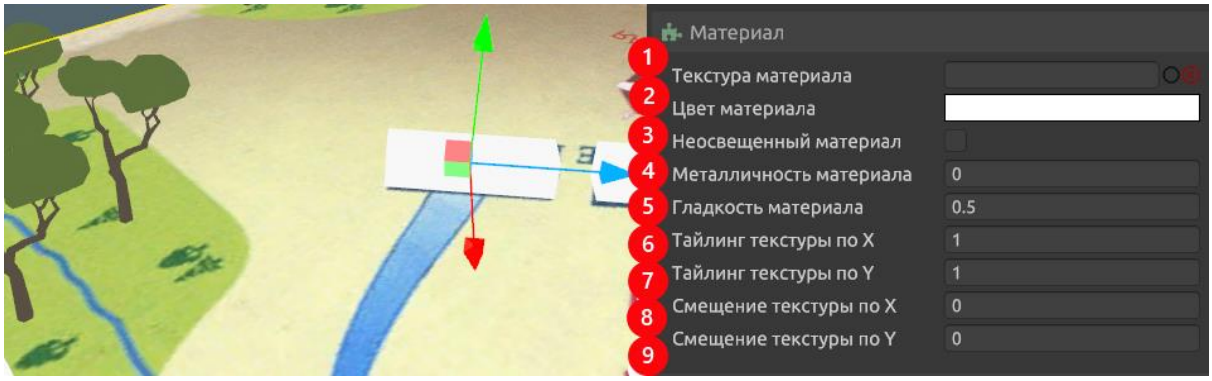
Все объекты, представленные на скриншоте ниже, являются примитивами и с их помощью нам нужно будет расставить основные объекты на локации в следующем проекте.

.

Далее давайте рассмотрим свойства этих примитивных объектов:

1. Текстура материала - ресурс в виде изображения, который можно применять к примитиву для изменения его визуальной составляющей.
2. Цвет материала - позволяет изменить цвет примитива.
3. Неосвещенный материал - активная галочка снимает действие всех эффектов света на данный объект.
4. Металличность материала - позволяет изменять степень похожести на металл у объектов.
5. Гладкость материала - влияет на отражающие физические свойства объекта
6. Тайлинг текстуры по X - изменяет степень плотности размещения текстуры по оси X
7. Тайлинг текстуры по Y - изменяет степень плотности размещения текстуры по оси Y
8. Смещение текстуры по X - сдвигает текстуру на объекте по оси X
9. Смещение текстуры по Y - сдвигает текстуру на объекте по оси Y

Свойства примитивов 5-9 в данном кейсе нам не понадобятся и мы рассмотрим их в дальнейших кейсах.



Самостоятельная работа: Обязательно попробуйте поработать со всеми этими свойствами и посмотрите на что они влияют. Попробуйте выставить разные параметры.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое текстура материала и какие текстуры использовать для примитивных объектов?
2. Что такое тайлинг и смещение текстуры?
3. Чем отличаются 2D примитивы от 3D?
4. Какие примитивные объекты существуют в XRMS Varwin?
5. Как свет влияет на 3D - объекты в сцене?

Занятие 4.2 Размещение примитивов на сцене

Цель:

Построить локацию образовательного проекта “Реконструкция сражения”, использовать элементы UI-дизайна. Подготовить локацию для применения логики.

Задачи:

- Разместить все необходимые объекты на локации
- Закрепить навык использования объектов “Текст”, как элементов UI-дизайна
- Научиться правильно использовать иерархию объектов в Varwin
- Расставить все объекты на локации структурировано, с использованием иерархии
- Усвоить навыки масштабирования, перемещения и поворота объектов.
- Усвоить навыки тестирования своих проектов на баги/ошибки
- Научиться настраивать текстуры на примитивных объектах
- Закрепить навыки использования свойств объекта
- Усвоить навык поиска референсов для локации и объектов на локации

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #5.

В этом кейсе мы познакомимся с примитивами Varwin (из библиотеки “Простые объекты”), а также научимся работать со стандартной логикой перемещения, вращения и масштабирования объектов.

Работая в привычной нам логике каждый проект начинаем с технического задания.

Техническое задание:

Необходимо разработать приложение на платформе Varwin, которая бы в схематичной форме представляла все фазы Куликовской битвы.

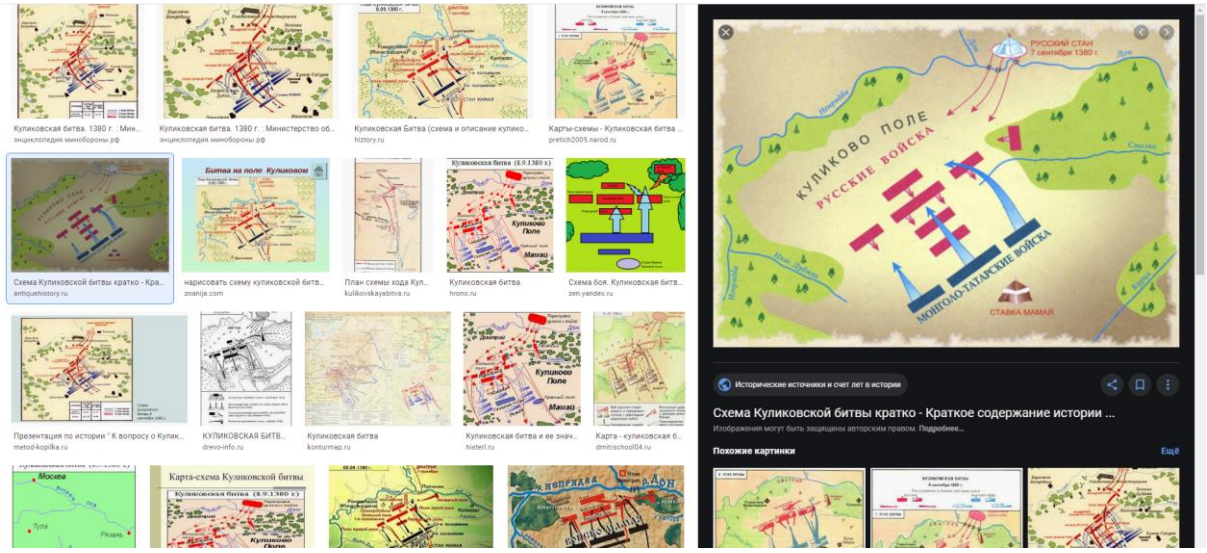
1. В приложении должна быть реализована механика перемещения схематичных войск.
2. Если полк несет потери, то его размер должен уменьшаться.
3. Необходимо использовать правдоподобную схему Куликовской битвы.
4. Приложение должно отражать три основные фазы боя:
    a. Первый этап составил бой авангардов: русских Сторожевого и Передового полка с легкой конницей Золотой Орды. Летопись указывает, что столкновение уже на этом этапе носило ожесточенный характер «и бысть брань крепка и сеча зла зело». Почти вся пехота этих полков «аки древеса сломишася, и аки сено посечено лежаху...»
    b. Одновременно конница Мамая атаковала полки Правой и Левой руки. Атака русского правого фланга была отбита. Более успешной была атака монгольской конницы против левого фланга русского войска. Почти все воеводы полка Левой руки были убиты. Полк стал подаваться назад, освобождая место для атакующей татарской конницы. Сражающиеся отошли до берега Непрядвы. Путь отхода к переправам был отрезан.
    c. Именно в это время воевода Дмитрий Боброк, наблюдавший из Зеленой Дубравы за ходом сражения, решил включить в него Засадный полк, состоящий из отборной, хорошо вооруженной конницы. Своевременный ввод в сражение крупного резерва, изменивший соотношение сил на направлении главного удара Орды, послужил поворотным моментом всего сражения. Ордынская конница пришла в смятение.

Создайте проект и выберите пустую сцену (Empty Scene), как в кейсовом задании с панорамами. В этом случае мы создадим локацию “с нуля”, используя средства платформы Varwin.

Создание локации

Для начала нам необходимо будет реализовать реалистичное поле боя, которое бы соответствовало реальным историческим событиям.

Для этого проще всего найти схему битвы в “Google” или “Яндекс” по запросу “Схема Куликовской битвы”. Таким образом, нам будет проще собирать логику приложения, в том числе. Т.к. все условные обозначения и перемещения войск уже есть на карте.



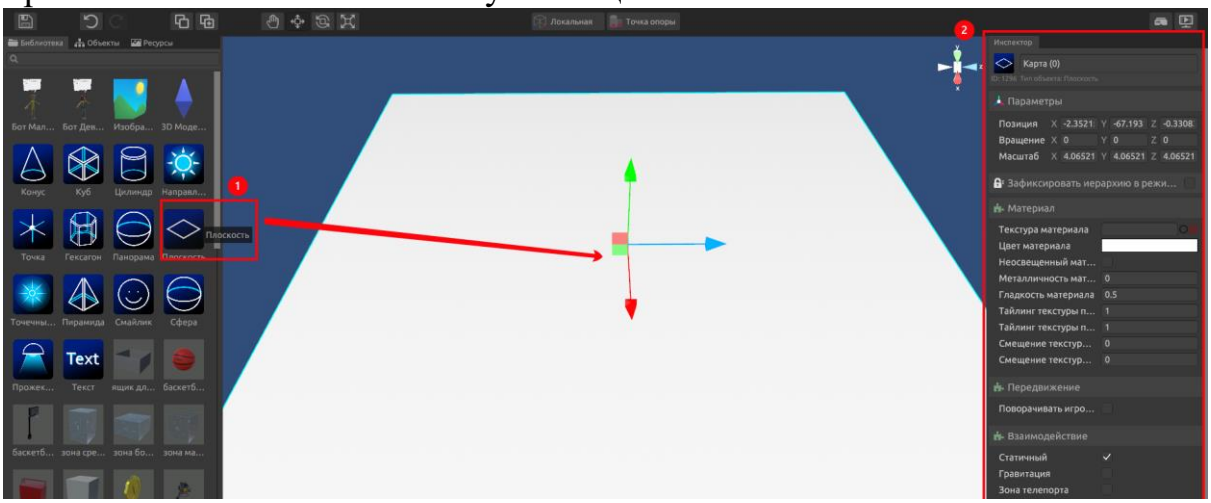
Далее загружаем понравившуюся картинку и импортируем ее в ресурсы Varwin.

***Совет**: если вы забыли как импортировать ресурсы, то вы всегда можете обратиться к главе 2.2.2!*

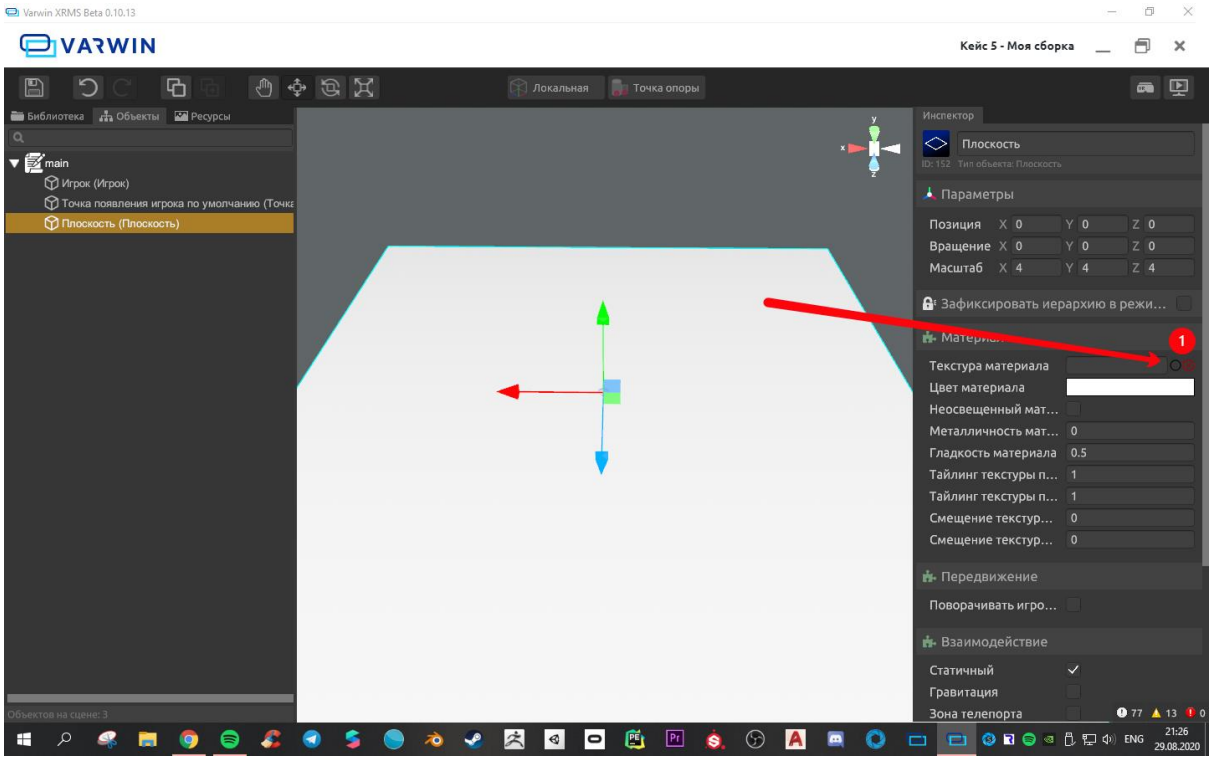
Далее необходимо сделать основу в виде “земли” на пустой сцене по которой будут перемещаться войска. Для этого выберете уже знакомый объект “Плоскость”, вытащите его на сцену (1) и не забудьте установить правильные свойства и параметры (2).

Не стесняйтесь сразу сделать этот объект побольше, т.к. поле боя должно быть большим относительно игрока.

А также оно обязательно должно являться препятствием , чтобы другие объекты не проваливались сквозь него и не иметь гравитации, чтобы объект не провалился в бесконечность пустой сцены.



Далее мы применяем (1) загруженный ресурс “Схема Куликовской битвы” как текстуру для объекта “Плоскость”. У нас получается поле битвы в двухмерном виде, поверх этого вида мы будем выстраивать дальнейшие взаимодействия.



Следующее что нам нужно сделать это разместить сферическую панораму облаков, т.к. смотреть реконструкцию боя мы будем сверху вниз и как бы находится в облаках. Вспоминаем второй кейс, ищем ресурс для панорамы, добавляем объект панорама на сцену и масштабируем до нужных нам размеров, чтобы закрыть края плоскости.

То что мы сделали с панорамой называется скайбокс**.**

Определение:

Скайбо́кс *(англ. sky — «небо» и box — «коробка») — объект в трёхмерной графике, играющий роль неба и горизонта. Представляет собой несложную трёхмерную модель (как правило, куб или сферу), с внутренней стороны которой натянута текстура неба (так называемая «кубическая текстура»).*

Для самостоятельной работы вам предлагается расставить дополнительные объекты на локации, так называемые “пропсы”.

Определение: *Объекты которые раскрывают или дополняют локацию в дизайне уровней называют «**пропсами**» (от англ. «props» — реквизит). Они выполняют роль бутафорского реквизита на помогают зрителю в полной мере прочувствовать обстановку места действия.*

Обычно при планировании нового уровня выделяют три группы пропсов:

Крупногабаритные объекты (массивные конструкции, павильоны, ограждения, автомобили, грузовики, деревья и т.д.)

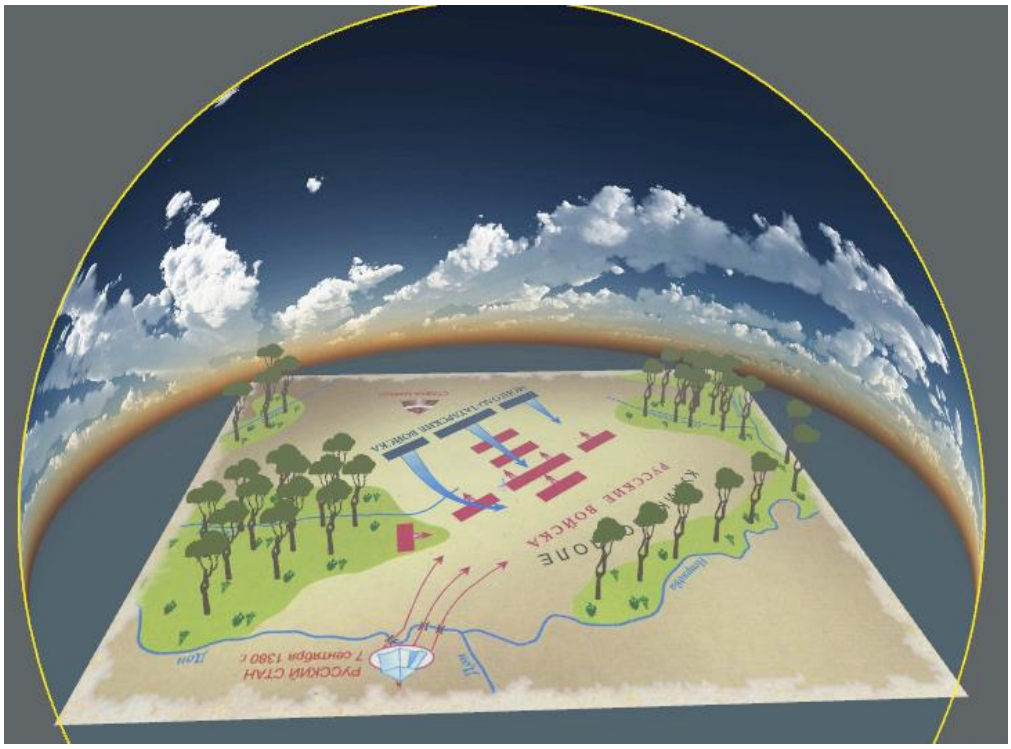
Среднегабаритные объекты (домашняя и уличная мебель, рекламные вывески, антенны, мачты освещения, контейнеры и т.п.)

Малогабаритные объекты (мелкие детали интерьера и экстерьера — бутылки, камни, трава, листья и т.д.)

Основными объектами на нашей локации будут примитивы, играющие роль сражающихся полков. А пропсами на нашей локации будут деревья.

Самостоятельная работа: ищем 3D-модели для пропсов и расставляем их на локации. Нужно максимально украсить локацию.

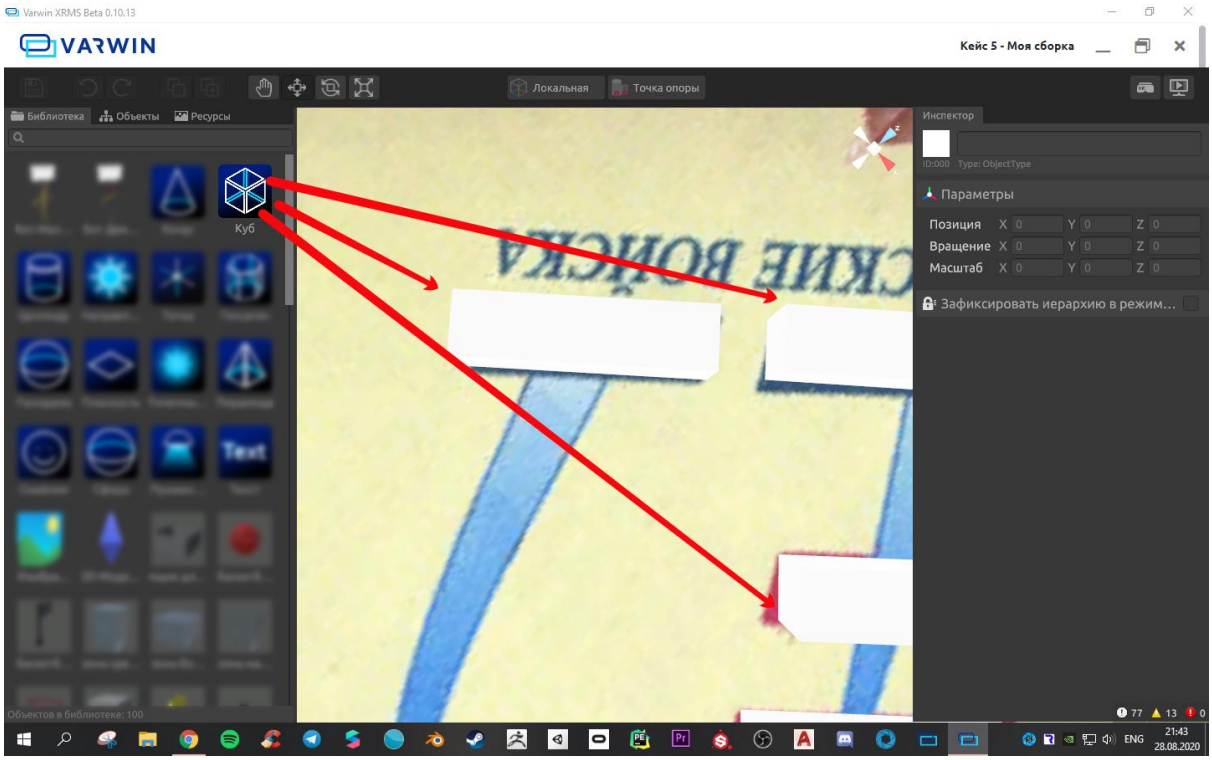
В конечном итоге у Вас должно получиться что-то подобное:



Работа с примитивами

В этом блоке мы с помощью примитивных объектов расставим войска и настроим их свойства.

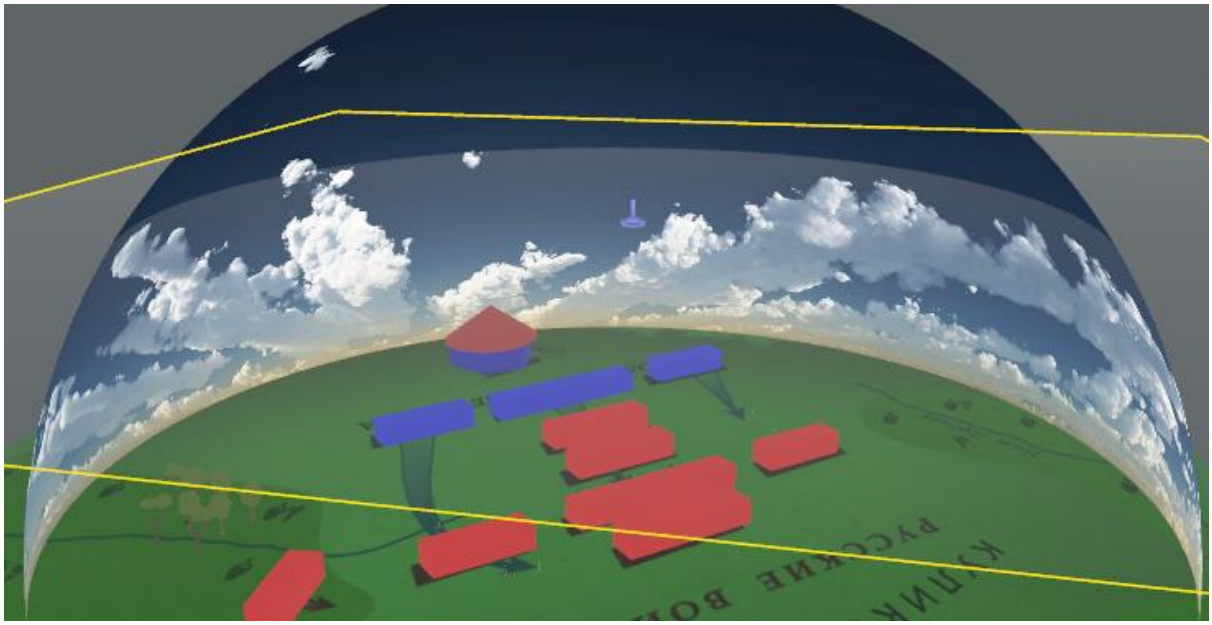
Далее согласно схеме боя мы расставляем войска, делаем это с помощью объектов типа “Куб” и масштабируем, вращаем и перемещаем объекты чтобы они смотрелись так как нам нужно.



Также обозначим с помощью других примитивов места базирования главнокомандующих, Ставку Мамая и Русский Стан.

Самостоятельная работа: Вам необходимо изменить цвет материала русских войск на красный, а цвет войск Мамая на синий.

Также необходимо создать дополнительную плоскость по которой можно будет телепортироваться и рассматривать поле боя. Она должна находиться на уровне облаков. Выглядеть это должно вот так:



Последним шагом расстановки объектов на сцене нам нужно создать несколько вспомогательных объектов типа “Точка”. В дальнейшем в редакторе логики “Blockly” мы создадим события с помощью которых наши войска будут

двигаться, и благодаря этим точкам нам будет проще задать направление движения.

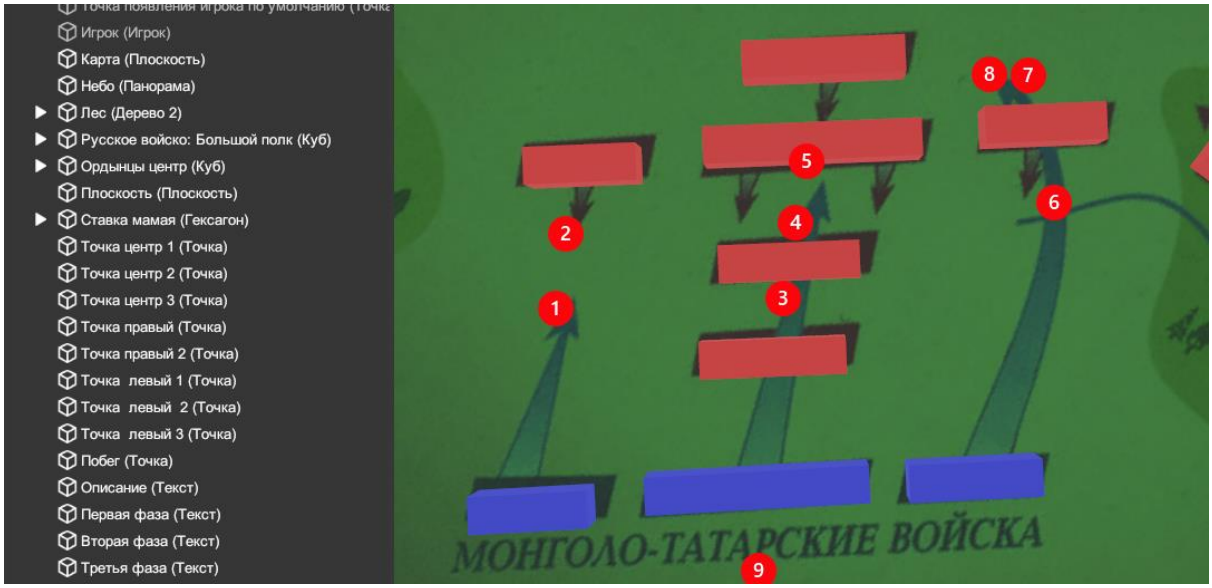
1 и 2 - точки правого фланга;

3, 4 и 5 - точки центра;

6, 7 и 8 - точки левого фланга;

9 - направление побега войск Мамая, её можно поставить далеко за нашей схемой боя.

По координате Y эти точки расположены выше на половину своей высоты, чтобы войска были расположены над плоскостью со схемой сражения.

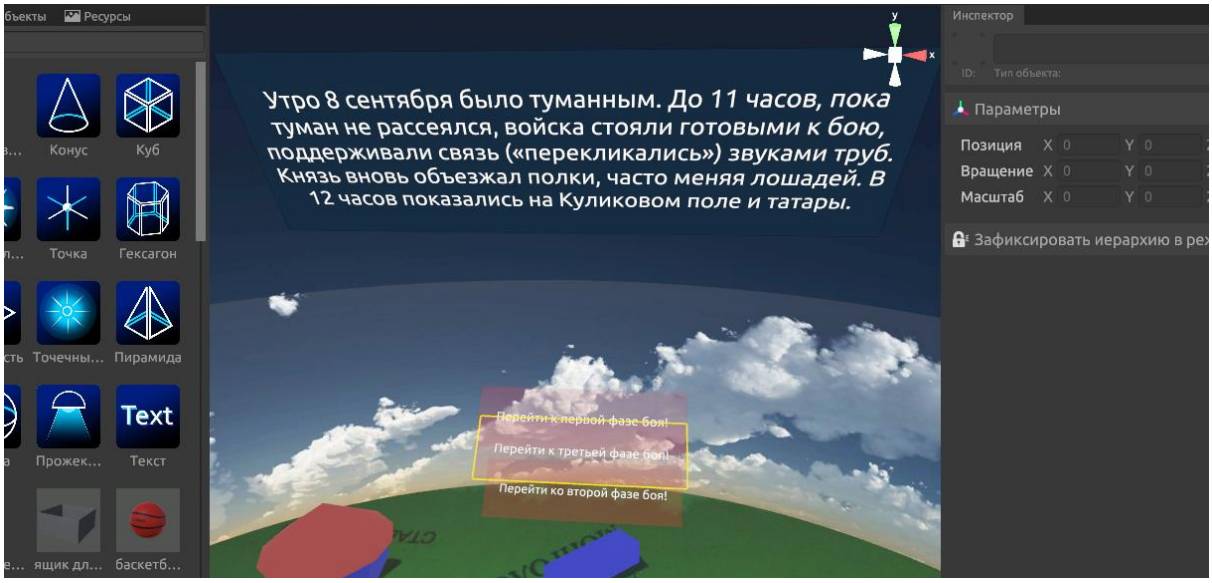


Создание UI-дизайна

Для того чтобы анимировать фазы боя нам необходимо разработать несколько кнопок для переключения между фазами, а также главную информационную панель, в которую мы будем добавлять описание определенной фазы из ТЗ.

Для кнопок будем использовать объект типа “Текст”. Мы уже делали подобные элементы во втором кейсе с панорамами, только там они использовались для переключения между панорамами, а тут будут использоваться для перехода из фазы в фазу и анимации.

Самостоятельное задание: создать понятный UI-дизайн переключения между фазами сражения. Должно быть 3 кнопки на каждую фазу, при переключении на каждую фазу должно присутствовать описание этой фазы. Финальный UI-дизайн должен выглядеть примерно так:



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое пропсы и какие они бывают?
2. Какие примитивные объекты вы использовали в этом проекте?
3. Что такое скайбокс?
4. Сколько объектов типа “Пропсы” можно использовать на одной локации?
5. Что такое иерархия, родительские и дочерние объекты?

Занятие 4.3 Стандартные логические блоки объектов Varwin

Цель:

Познакомиться со стандартными логическими блоками и событиями.

Задачи:

- Узнать что такое логические блоки в XRMS Varwin и какими функциями они обладают
- Узнать что такое события и как они работают
- Закрепить на практике умение работать со стандартной логикой и событиями в XRMS Varwin
- Сформировать понимание уместности использования блоков с событиями

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #4.

Сейчас нам необходимо рассмотреть блоки стандартной логики объектов для дальнейшего понимания возможностей редактора логики Blockly и как мы можем взаимодействовать с объектами. Эта таблица будет служить вам справкой при работе со стандартной логикой, работой со свойствами из редактора логики и стандартными событиями

№	Логический блок	Описание блока
	Действия	

1	Задать расстояние остановки перед целевым объектом **X (м)**	Устанавливает на каком расстоянии до этого объекта остановится другой, при перемещении к нему
2	Перемещаться в направлении **(X/Y/Z)** со скоростью **X (м/с)**	Объект будет перемещаться по определенной оси с определенной скоростью бесконечно или остановится при наличии дополнительного условия
3	Перемещаться в направлении **(X/Y/Z)** на расстояние **X (м)** со скоростью **X (м/с)** *(один раз/повторяясь/туда-сюда)*	Объект будет перемещаться по определенной оси на определенное расстояние (в метрах) с определенной скоростью (в метрах в сек.) либо один раз, либо туда-обратно, либо бесконечно.
4	Перемещаться в направлении **(X/Y/Z)** в течении **X (сек)** со скоростью **X (м/с)** *(один раз/повторяясь/туда-сюда)*	Объект будет перемещаться по определенной оси в течении определенного времени (в секундах) с определенной скоростью (в метрах в сек.) либо один раз, либо туда-обратно, либо бесконечно.
5	Перемещаться в сторону объекта **X** со скоростью **X (м/с)** не прекращая *(истина/ложь)*	Объект будет перемещаться в сторону определенного объекта с определенной скоростью (в метрах в сек.), предмет будет перемещаться в сторону указанного объекта постоянно, если другой объект перемещается (истина) или один раз (ложь).
6	Перемещаться по маршруту **X** со скоростью **X (м/с)**	Объект будет перемещаться по маршруту с определенной скоростью (в метрах в сек.). Чтобы создать маршрут нужно сделать список из объектов по координатам которых будет двигаться объект.
7	Приостановить перемещение	Перемещение останавливается при определенном условии
8	Задать минимальный угол поворота к объекту **X (градусов)**	Устанавливает на каком угле к этому объекту остановится другой, при повороте
9	Вращаться вокруг оси **(X/Y/Z)** в течении **X (сек)** со скоростью **X (град/с)** *(один раз/повторяясь/туда-сюда)*	Объект будет вращаться вокруг определенной оси с определенной скоростью и в течение X секунд либо один раз, либо туда-обратно, либо бесконечно.
10	Вращаться вокруг оси **(X/Y/Z)** на **X (градусов)**	Объект будет вращаться вокруг определенной оси с определенной

	со скоростью **X (град/с)** *(один раз/повторяясь/туда-сюда)*	скоростью на X градусов либо один раз, либо туда-обратно, либо бесконечно.
11	Вращаться вокруг оси **(X/Y/Z)** со скоростью **X (град/с)**	Объект будет вращаться вокруг определенной оси в течении определенного времени с определенной скоростью бесконечно или остановится при наличии дополнительного условия
12	Повернуться к объекту **X** со скоростью **X (град/с)**	Объект будет вращаться с определенной скоростью пока не достигнет угла поворота направленного на целевой объект.
13	Приостановить/возобновить/остановить вращение	Вращение останавливается при определенном условии
14	Приостановить/возобновить/остановить масштабирование	Масштабирование останавливается при определенном условии
15	Масштабировать в **X раз** по оси **(X/Y/Z)** в течении **X (сек)** *(один раз/повторяясь/туда-сюда)*	Объект будет увеличиваться/уменьшаться в X раз по оси X/Y/Z в течении X секунд, либо один раз, либо туда-обратно, либо бесконечно.
16	Масштабировать со скоростью **X (м/с)** по оси **(X/Y/Z)** в течении **X (сек)** *(один раз/повторяясь/туда-сюда)*	Объект будет увеличиваться/уменьшаться в X раз по оси X/Y/Z в течении X секунд со скоростью X, либо один раз, либо туда-обратно, либо бесконечно.
17	Масштабировать в **X** раз по оси **(X/Y/Z)**	Объект будет увеличиваться/уменьшаться в X раз по оси X/Y/Z
18	Активировать/Деактивировать и Включить/Выключить	Показать или скрыть объект в режиме просмотра при определенном условии
Переменные/характеристики объекта		
1	Статичный объект (истина/ложь)	Данные свойства объектов мы уже разбирали в предыдущих кейсах. Единственное знание, которое нужно здесь усвоить это то, что можно включать/выключать данные свойства через редактор логики Blockly, при выполнении определенных условий.
2	Гравитация включена	
3	Является зоной телепорта	
4	Является препятствием	
5	Масса объекта	
6	Пружинистость объекта	

7	Можно брать в руку	
8	Можно использовать	
9	Можно дотронуться	
	События	
1	Объект взят в руку	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект взят в руку.
2	Объект отпущен из руки	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект отпущен из руки.
3	Объект начали использовать	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект начали использовать, то есть при нажатии на соответствующую кнопку.
4	Объект закончили использовать	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект закончили использовать, то есть при отпускании соответствующей кнопки.
5	До объекта дотронулись	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект руки игрока находятся рядом или внутри целевого объекта.
6	Объект прекратили трогать	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что объект руки игрока вышел из целевого объекта.
7	Началось столкновение	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что два объекта (один из которых является целевым) столкнулись с друг другом, происходит коллизия.
8	Столкновение закончилось	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что два объекта (один из которых является целевым) закончили столкновение с друг другом, коллизия закончилась.

9	Объект попал внутрь целевого объекта	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что два объекта (один из которых является целевым) начали пересекаться при выполнении проекта. Может выполниться только если объект НЕ является препятствием.
10	Объект оказался снаружи целевого объекта	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что два объекта (один из которых является целевым) закончили пересекаться при выполнении проекта. Может выполниться только если объект НЕ является препятствием.
11	Движение завершено	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что движение объекта завершилось.
12	Целевой объект достигнут	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что целевой объект достигнут. Если объектов несколько, то указываем каждый внутри события.
13	Точка пути достигнута	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что точка пути, заранее обозначенная целевой достигнута (ID). ID - это нумерация объекта в списке (с нуля).
14	Вращение завершено	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что вращение объекта завершилось.
15	Поворот к объекту завершен	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что вращение в сторону целевого объекта завершилось.
16	Масштабирование завершено	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия что масштабирование объекта завершилось.

На этом этапе обучающиеся применяют полученные ранее знания и тестируют стандартную логику и события в свободном формате. Главная задача попробовать как можно больше блоков из редактора логики Blockly.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие стандартные взаимодействия между игроком и объектом можно настраивать в Varwin?
2. Что такое «расстояние остановки перед целевым объектом», для чего оно используется?
3. Что такое “target” и как она используется в событиях?

Занятие 4.4 Сборка логики из стандартных логических блоков

Цель:

Разработать логику для проекта “Реконструкция сражения” и научиться использовать стандартные логические блоки для примитивов.

Задачи:

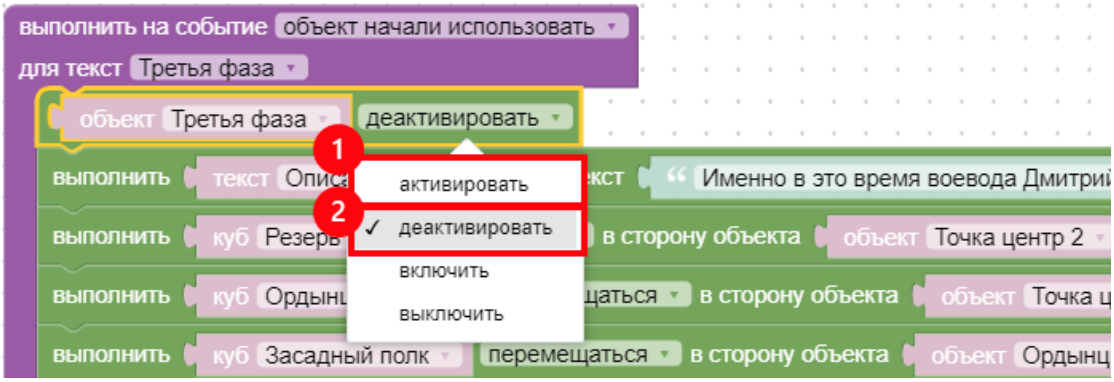
- Усвоить навык правильного использования иерархии объектов в Varwin
- Закрепить навыки масштабирования, перемещения и поворота объектов в редакторе логики.
- Усвоить навыки тестирования своих проектов на баги/ошибки
- Закрепить навыки использования свойств объекта
- Реализовать полноценную логику проекта согласно техническому заданию
- Структурировать логические блоки в редакторе логики, использую разные вкладки

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #4.

Создание логики

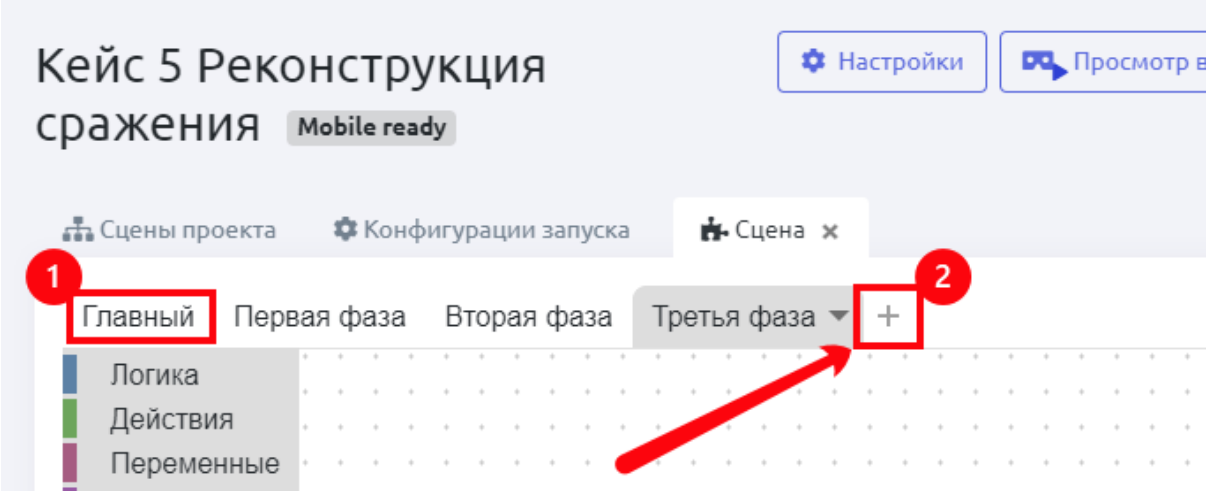
Очень важное новое знание редактора логики, которое нам нужно усвоить в этом кейсовом задании - это активация (1) и деактивация (2) объектов.



У каждого объекта в XRMS Varwin есть параметры активации и деактивации. С помощью них мы можем контролировать видимость объектов на сцене. Если активировать объект, то он будет виден на сцене. Если деактивировать, то объект перестанет быть видим.

Вкладки редактора логики

Чтобы нам было удобно работать с логикой в XRMS Varwin специально созданы удобные вкладки редактора логики (1). С помощью этих вкладок мы можем разнести громоздкие конструкции логики в разные блоки, отвечающие за определенную область функционала приложения.

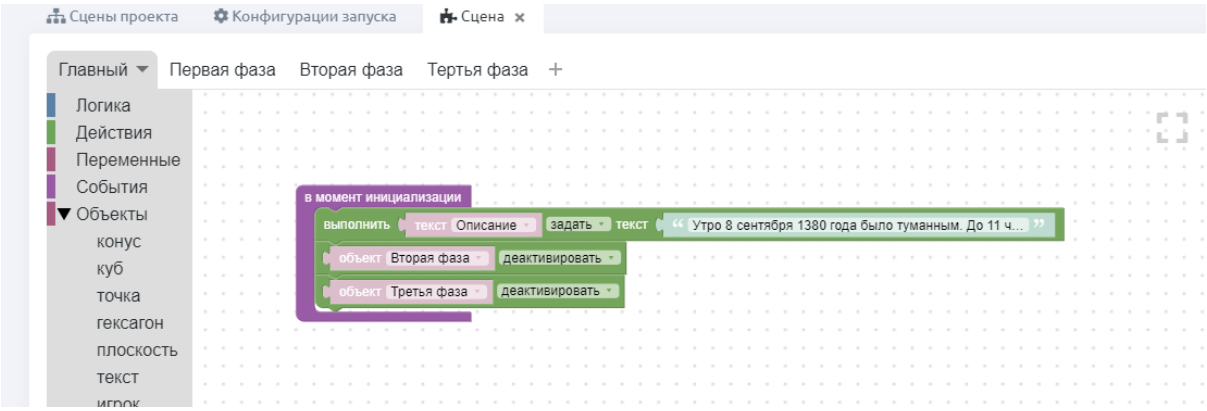


Чтобы создать новую вкладку необходимо нажать на (2) значок “+ и назвать её. Для данного проекта будет удобно *создать четыре вкладки*, как на скриншоте выше.

Создайте отдельную вкладку для каждой фазы боя.

После этого сразу проработаем вкладку редактора логики “Главный”. Нам необходимо задать описание для объекта “Текст-описание” и деактивировать кнопки перехода на вторую и третью фазу.

Всё это нам необходимо добавить в событие “в момент инициализации”. Подробное описание события можно посмотреть в таблице ниже.



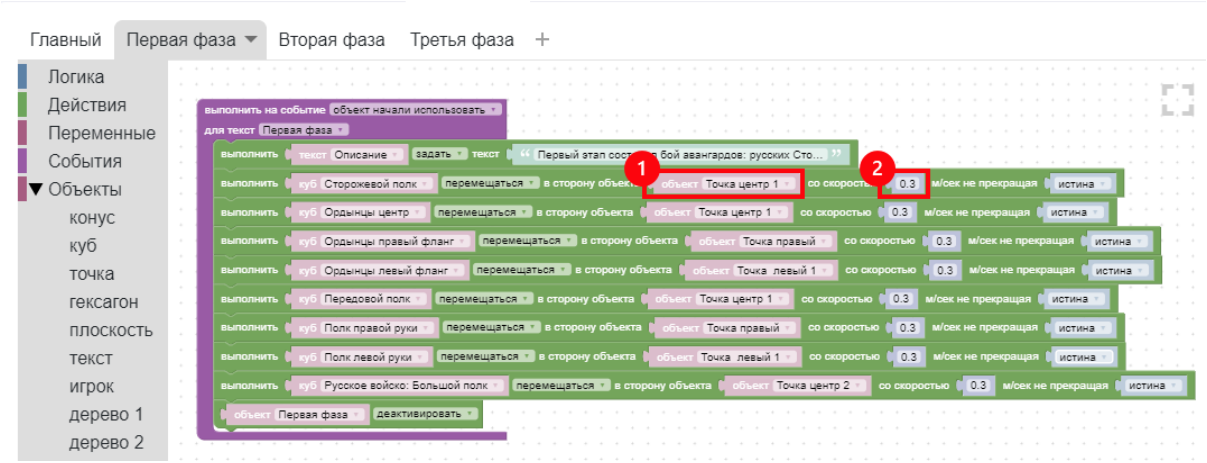
Сборка логики первой фазы

Теперь перейдем к вкладке редактора логики “Первая фаза”.Здесь нам необходимо задать алгоритм действий для события использования объекта “Текст-Первая фаза”.

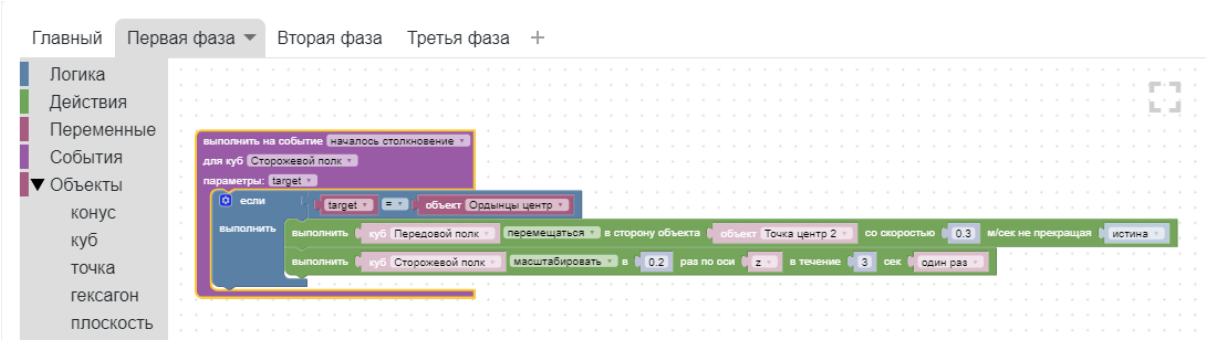
При взаимодействий с кнопкой “Первая фаза” описание должно изменяться на описание первой фазы сражения. И все боевые единицы должны перемещаться соответственно первой фазе сражения. Обязательно нужно выбрать точку (1) к которой будет перемещаться определенный полк и скорость (2) с которой он будет это делать.

Совет: *в этом кейсе как никогда важно тестировать приложение в процессе его разработки, чтобы удостовериться, что все полки двигаются так, как нам необходимо. Чтобы определить оптимальную скорость и направление перемещения периодически запускайте проект в Desktop-режиме из Desktop-редактора. Чтобы не загружать редактор каждый раз заново, при сборке логики просто сворачивайте окно Desktop-редактора.*

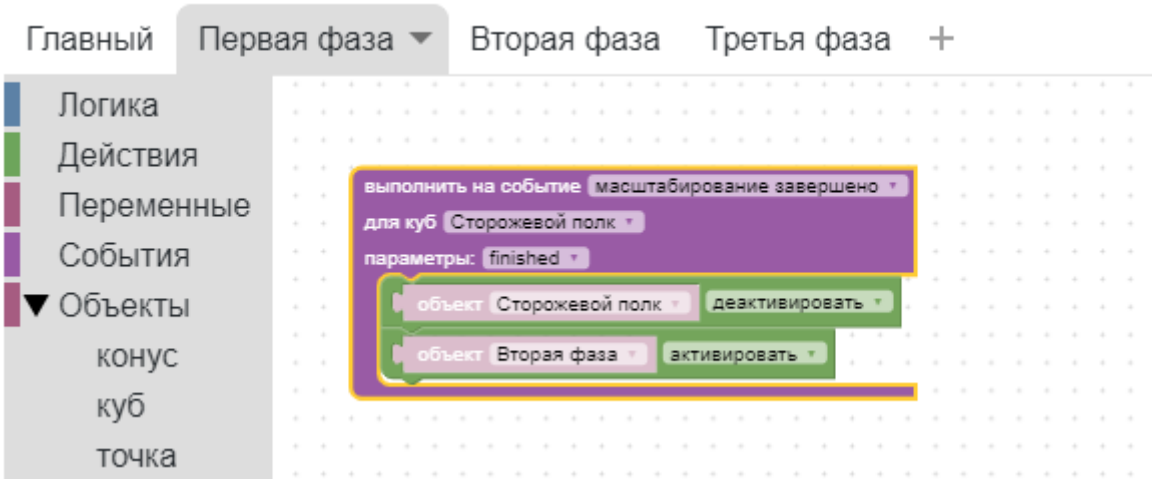
После всех перемещений объект “Текст-Первая фаза” должен быть деактивирован.



Также, при определении столкновения между объектами “Ордынцы центр” и “Сторожевой полк” должны происходить дополнительные перемещения и масштабирование блоков.



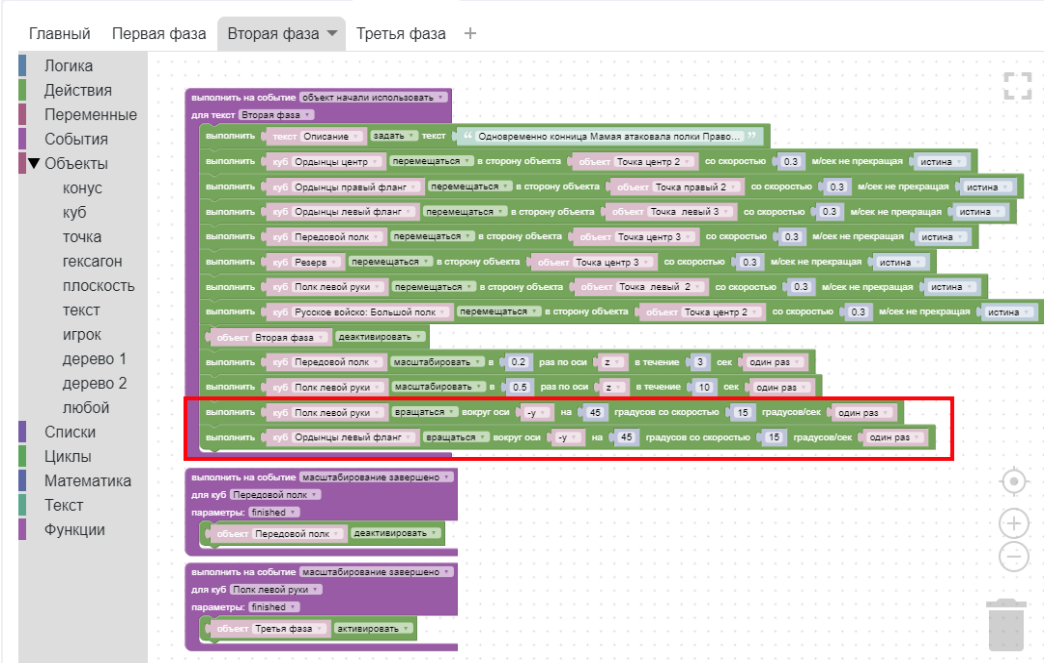
И как только масштабирование (то же самое что уничтожение полка) будет завершено нам необходимо убрать его с поля сражения и активировать кнопку “Вторая фаза”.



Сборка логики второй фазы

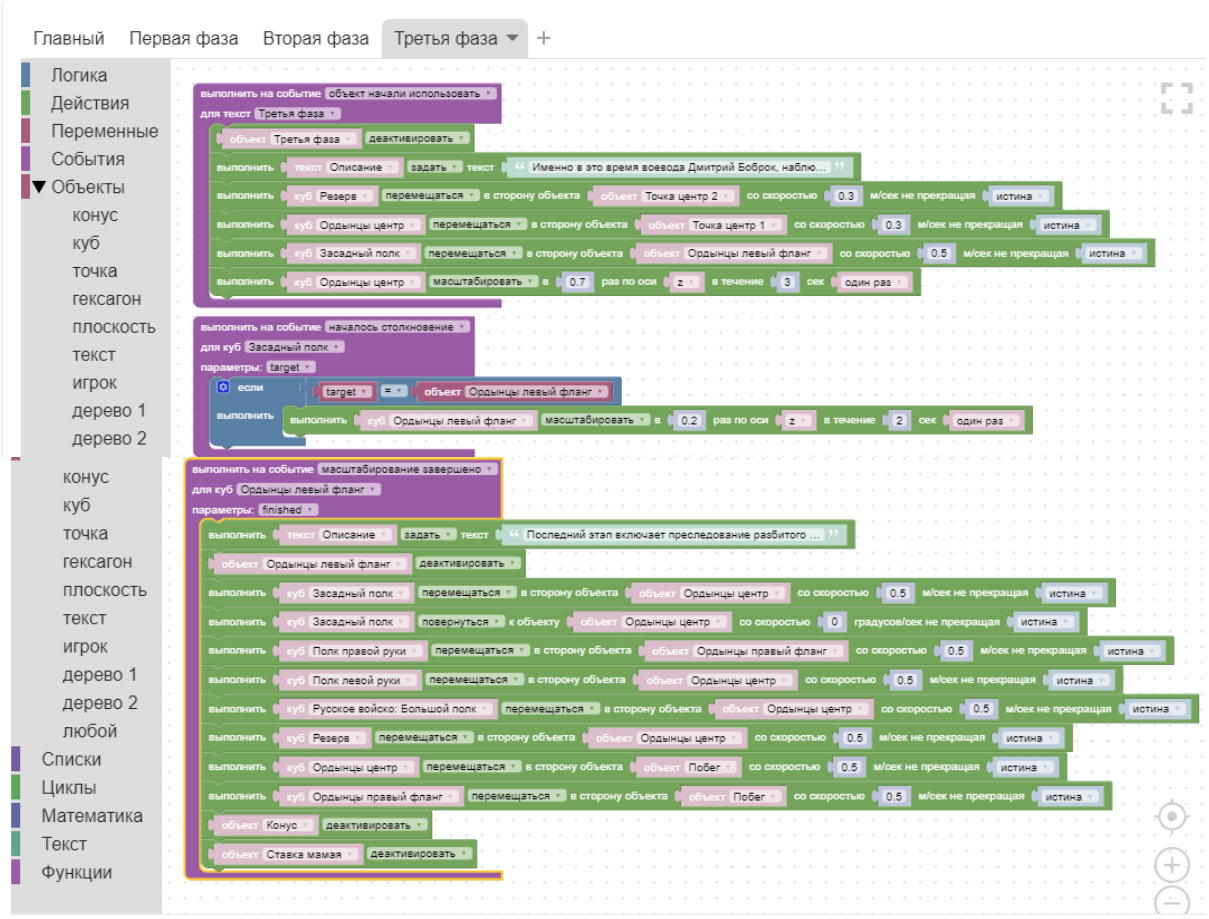
Во второй фазе у нас добавляется дополнительное действие позволяющее вращать военные полка. Логика остается такой же, нам нужно переместить объекты согласно тому как это происходило на поле сражения.

Чтобы заставить наши примитивы вращаться нам необходимо выбрать ось вдоль которой они будут вращаться, задать угол и скорость поворота.

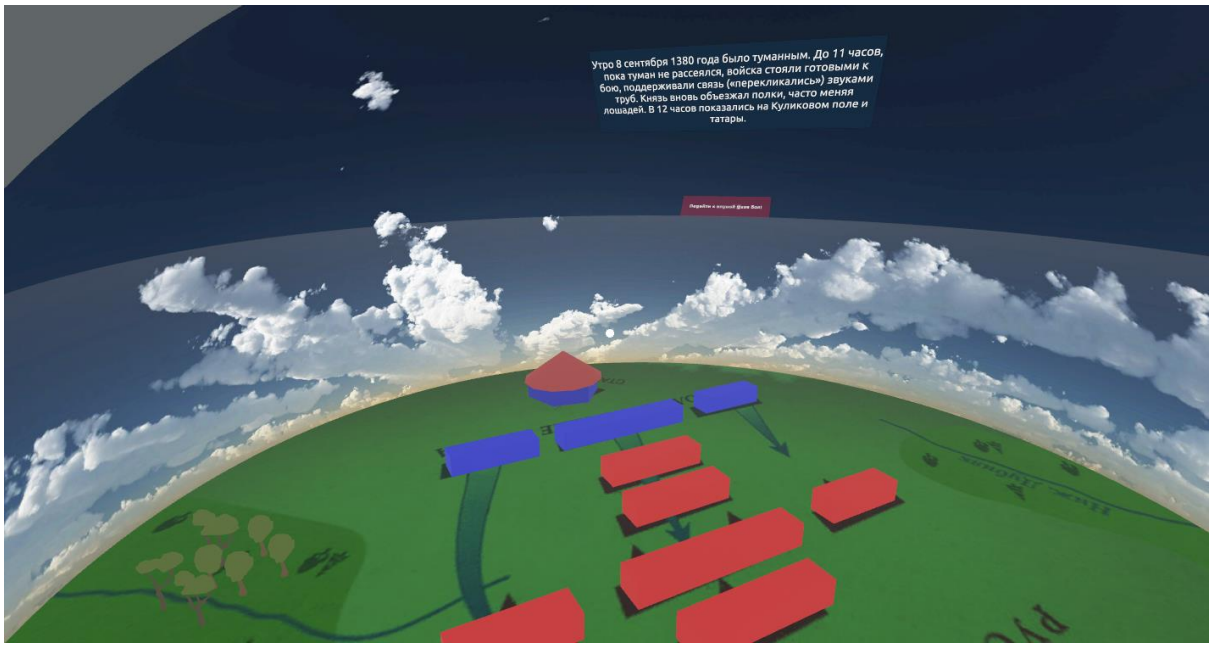


Сборка логики третьей фазы

Самостоятельно подумайте, как войска должны перемещаться в третьей фазе и что должно происходить на сцене. Финальная логика должна выглядеть так:



Финальный вариант размещения объектов на сцене выглядит так:



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие стандартные события существуют для примитивов Varwin?
2. Можно ли назвать примитивным объект “точка”?
3. Что такое активация и деактивация объектов?

4. Что такое вкладки редактора логики и для чего они нужны?

Занятие 4.5 Создание проекта "Молекулы"

Цель:

Разработать проект в котором мы объединяем различные атомы (в виде примитивных объектов) для получения химических соединений. Усвоение навыков, полученных в ходе практических занятий.

Задачи:

- Закрепить навык работы с примитивными объектами в Varwin.
- Закрепить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Закрепить навык работы с логикой базовых блоков в Blockly
- Закрепить навык работы с desktop - редактором XRMS Varwin
- Усвоить навыки работы с локацией и размещением объектов на сцене
- Сформировать понимание правильности использования дизайна интерфейсов, основанном на удобстве для конечного пользователя
- Закрепить навык работы с техническим заданием на проект

Формат: Самостоятельная работа, решение кейсового задания.

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

На этом этапе нужно усвоить навык формирования и фиксации технического задания.

Шаблон технического задания:

1. Тематика/направленность проекта;
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы или же свои объекты);
3. Основной функционал приложения;
4. Дополнительные возможности приложения;
5. План-схема сцен, которые будут реализованы;
6. Каким образом будет реализован, удобный для пользователя, UX/UI - дизайн;
7. Какую полезную функцию для общества несет проект, или будет нести при дальнейшей его доработке.

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Кейс:

Необходимо разработать проект, позволяющий изучать процесс формирования химических соединений. Важно показать молекулы отдельно и то, что они состоят из атомов двух или более различных химических элементов. Проект предназначен для использования на уроках химии.

Дополнительное задание, если позволяет время: реализовать дополнительные логические конструкции, которые должны быть зафиксированы в ТЗ. Например, показать привязку в таблице Менделеева и реализовать возможность выбора элементов прямо из таблицы Менделеева.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта
2. Нарисовать план расположения объектов на сцене
3. Зафиксировать дополнительные функции, которые будут реализованы в проекте.
4. В приложении обязательно должны использоваться примитивные объекты.
5. Должна быть задействована логики в редакторе логики для изменения параметров перемещения/вращения/масштабирования примитивных объектов.
6. Должно быть реализовано как минимум ТРИ химических соединения.

5 Функции

Занятие 5.1 Построение локации для образовательного проекта "Правила дорожного движения"

Цель:

Построить локацию образовательного проекта "Правила дорожного движения". Подготовить локацию для применения логики.

Задачи:

- Разместить все необходимые объекты на локации
- Научиться правильно использовать иерархию объектов в Varwin
- Расставить все объекты на локации структурировано, с использованием иерархии
- Усвоить навыки масштабирования, перемещения и поворота объектов.
- Усвоить навыки тестирования своих проектов на баги/ошибки
- Закрепить навыки использования свойств объекта
- Усвоить навык использования объекта “Зона”
- Изучить типы объектов освещения в Varwin

Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #6.

Техническое задание

Необходимо разработать приложение, обучающее правилам перехода через дорогу по пешеходному переходу.

Специальные требования:

1. Разработать кольцевую дорогу, по которой будет перемещаться объект “Спортивная машина”
2. Реализовать пешеходный переход, с помощью которого игрок будет изучать правила перехода через дорогу.
3. Реализовать светофор со всеми его функциями
    a. Обязательно разработать функцию переключения цветов светофора
    b. Цвета светофора должны переключаться по заданному времени (желтый - длится 3 секунды, красный и зеленый - длятся 5 секунд)
    c. Сделать панель для вывода времени на переход дороги
4. Реализовать логику движения машины по маршруту и корректность поворотов.
5. Реализовать логику задания
    a. Вспомогательные зоны запрета прохода
    b. Вспомогательная зона пешеходного перехода
    c. Вспомогательная зона выигрыша
6. Реализовать условие выигрыша

Создайте проект и выберите сцену “Город”, на ней уже есть зона телепорта и мы можем перемещаться по всей сцене. Это то, что нам нужно.

Собираем участок дороги

В этом разделе нам нужно вспомнить что мы делали в прошлом кейсе и в поработать с примитивами. С помощью плоскостей, изменения их материалов и масштабирования создадим один участок дороги.

Самое главное что нам необходимо сделать - это соблюдать правильность иерархии для того чтобы в дальнейшем нам было максимально удобно использовать объекты на сцене.

Определение: Иерархия объектов *– это специальный интерфейс для просмотра и работы с разветвленными структурами. Источником данных для иерархии являются хранимые объекты, поэтому в иерархию может быть заложена любая логика связи между родительским и дочерними объектами. Чаще всего иерархии используются для работы с задачами, но в общем случае иерархии могут отображать любые объекты (например, орг. структуру, состав вложенных групп и т.п.).*

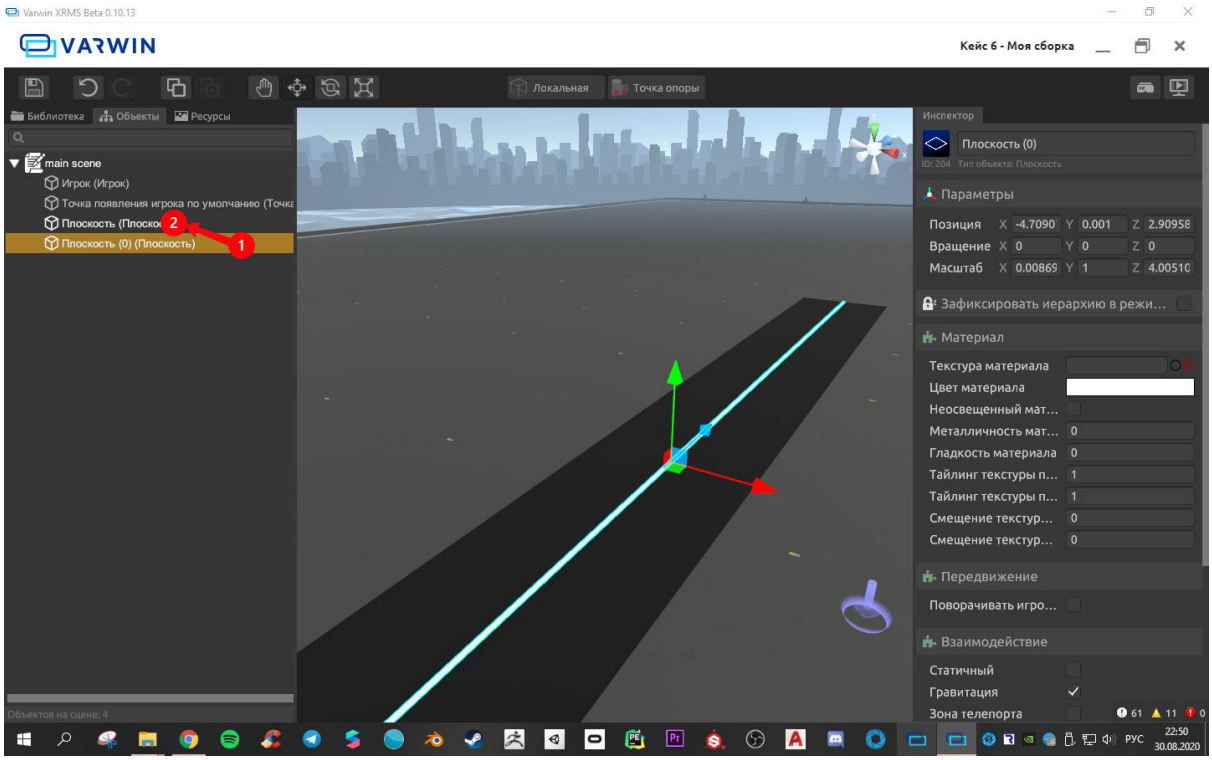
Для описания взаимосвязей между объектами в Varwin применяется определенная терминология.

Родительским (parent) *называется объект, который управляет связанными с ним вторичными, или дочерними, объектами.*

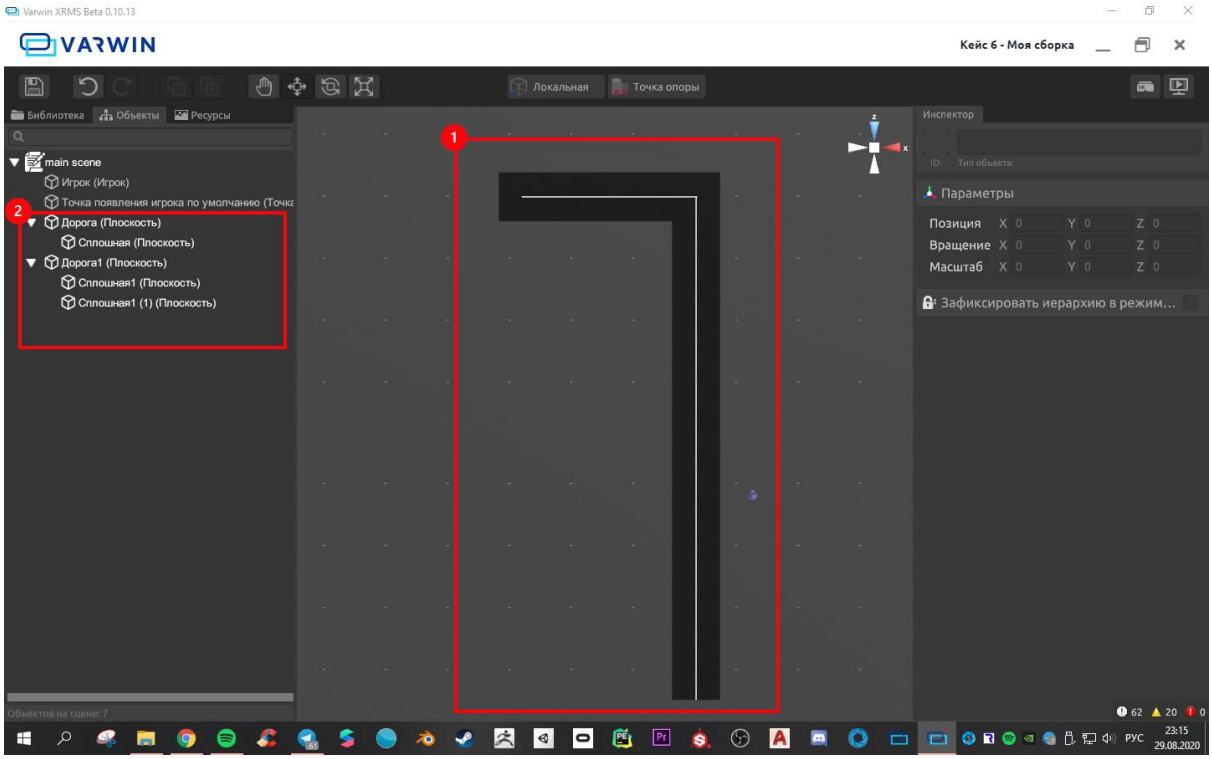
Дочерним (child) *называется объект, связанный с родительским и находящийся под его управлением. Один родительский объект может иметь много дочерних, однако у всякого дочернего объекта может быть только один родитель. Кроме того, один и тот же объект может быть родительским для одних объектов и в то же время дочерним - для других (многоуровневая иерархия).*

Давайте посмотрим на примере. Чтобы объединить два объекта и сделать их родительским и дочерним нужно перетащить один объект под другой во вкладке “Объекты”.

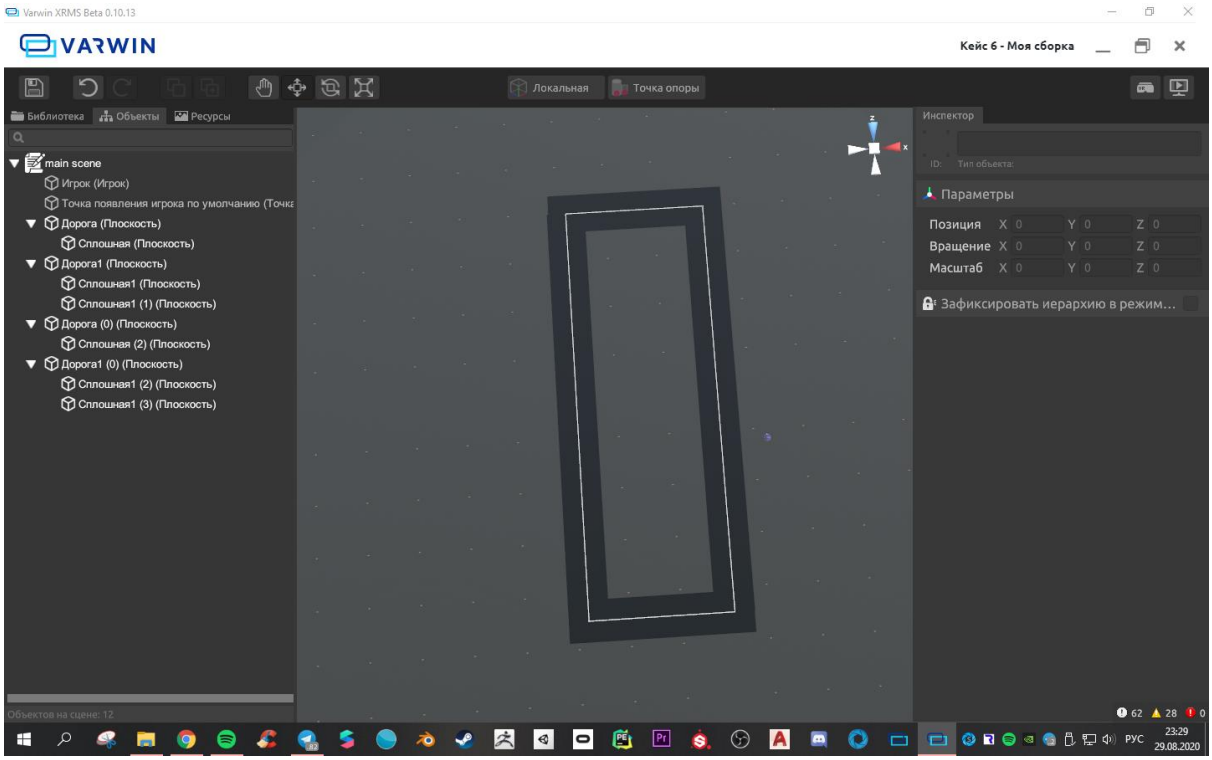
Примечание: В свойствах объекта существует параметр “Зафиксировать иерархию в режиме просмотра”. Что это означает? Допустим, мы захотим взять в руки какой-либо объект у которого есть несколько дочерних объектов, если галочка стоит то они будут двигаться как единое целое. Но стоит помнить что брать в таком случае необходимо родительский объект.



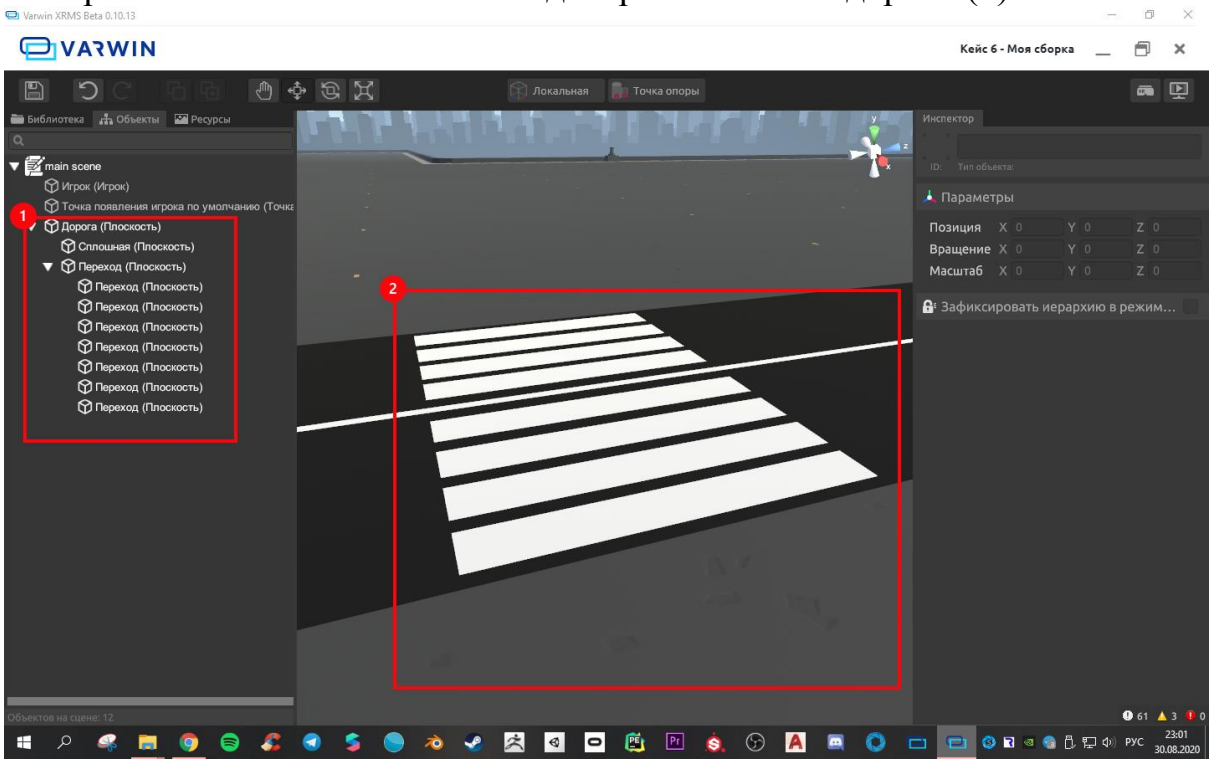
Теперь создадим с помощью плоскостей и материалов вот такой участок дороги (1) с правильной иерархией (2):



Далее, благодаря правильной иерархии, мы можем взаимодействовать и использовать дорогу и сплошную как один цельный объект. Давайте скопируем этот участок дороги и замкнем его по кругу.

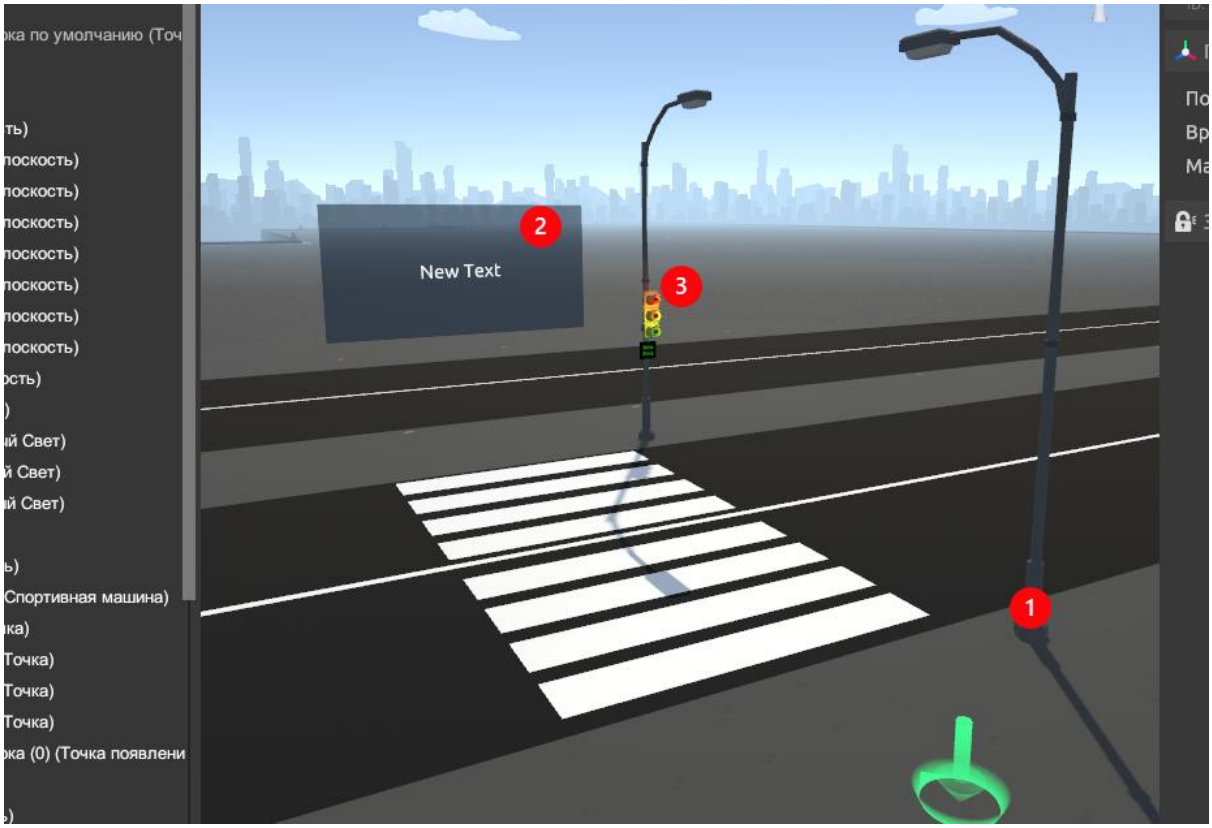


Осталось только с помощью плоскостей создать пешеходный переход (2) и закрепить все его элементы как дочерние объекты дороги (1).



Оборудуем пешеходный переход

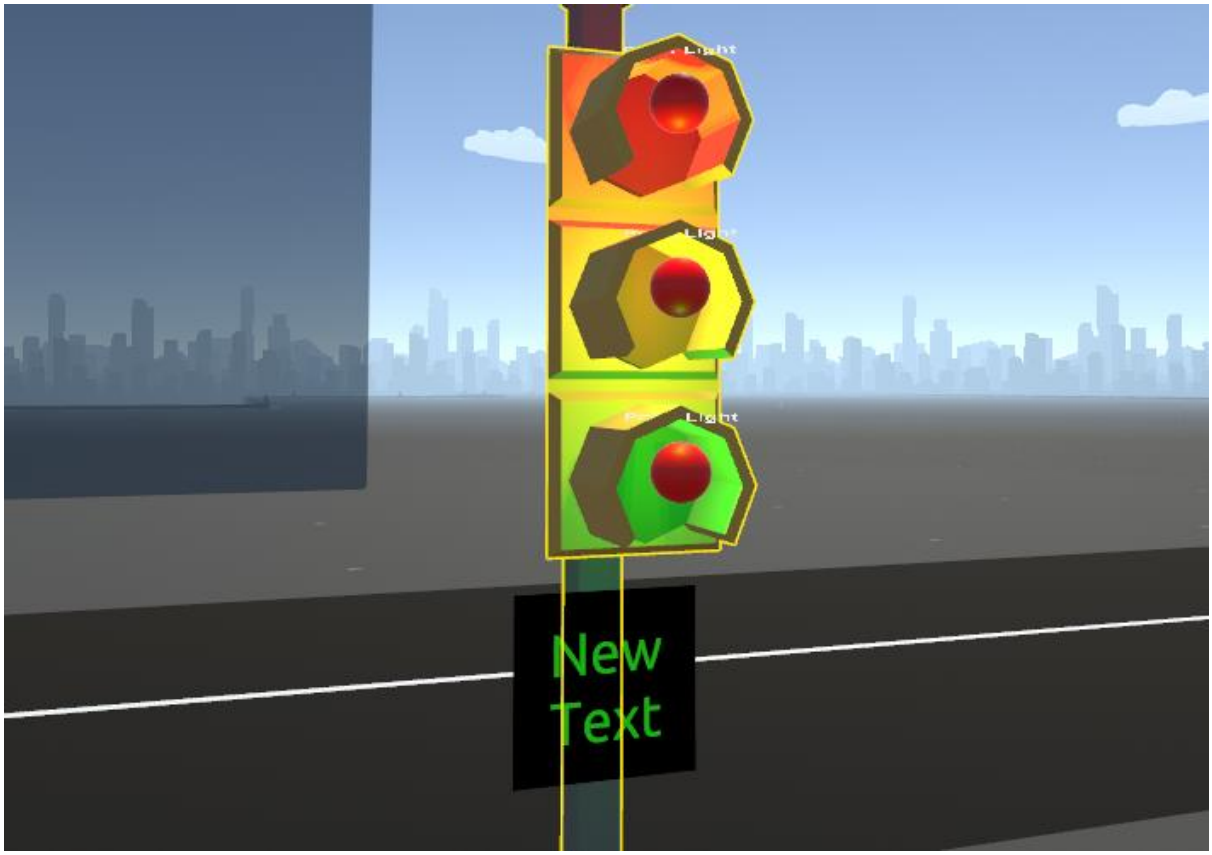
Для начала давайте создадим окружение и поставим пару фонарных столбов (1) по разные стороны дороги. Создадим два объекта “Текст” (2), которые будут элементами UI-дизайна. Один будет использоваться как текстовая панель с комментариями происходящего, второй для таймера под светофором. И используем объект “Светофор” (3) из пакета объектов “Мегаполис”.



Далее нам нужно создать эффекты свечения для светофора, но для начала давайте разберемся какие типы источников света есть в Varwin, и чем они отличаются.

Точечный свет	**Прожекторный свет**	**Направленный свет**
Точечный источник света излучает свет одинаково во всех направлениях, как лампочка.	Прожекторный источник света светит из точки в некотором направлении и освещает объекты только внутри конуса, как автомобильные фары.	Направленный свет источник света расположен бесконечно далеко и оказывает влияние на все объекты сцены. Эффект схож с солнечным светом.

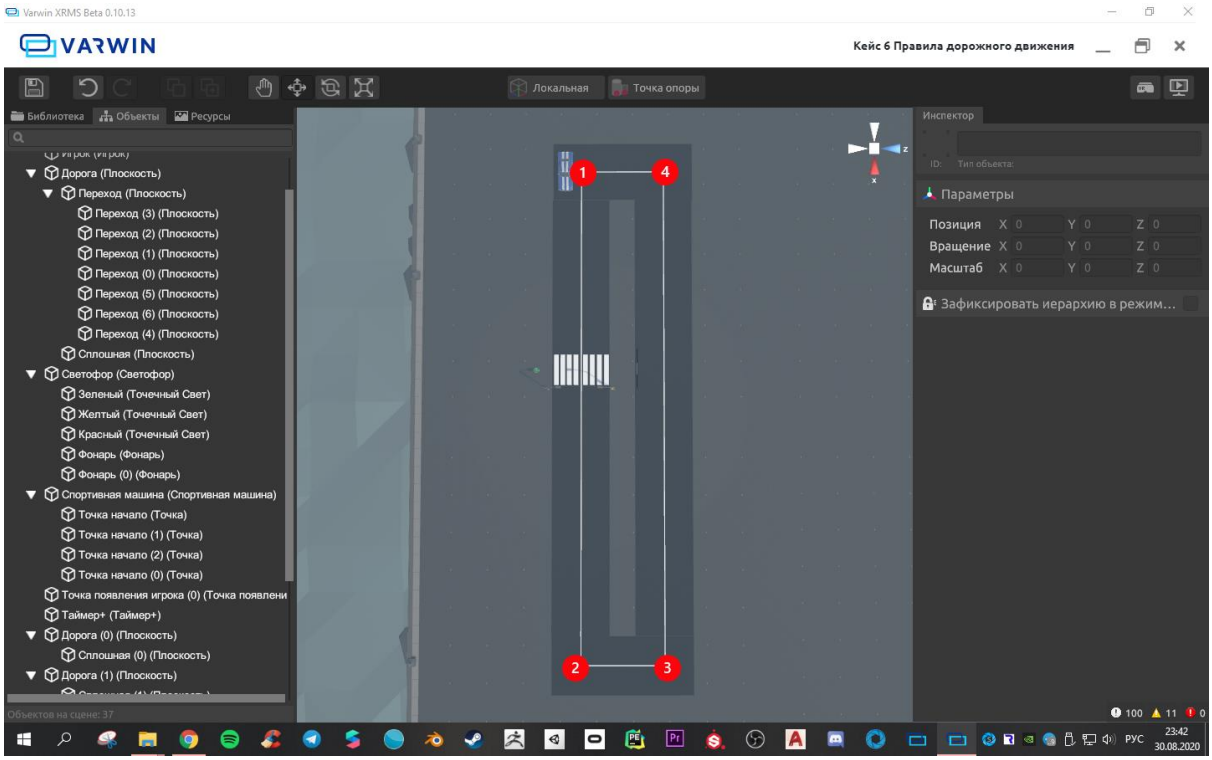
В светофоре мы будем использовать точечный свет. Давайте создадим три таких объекта и зададим им разные цвета в свойствах. Зеленый, желтый и красный.



Собираем маршрут для передвижения машины

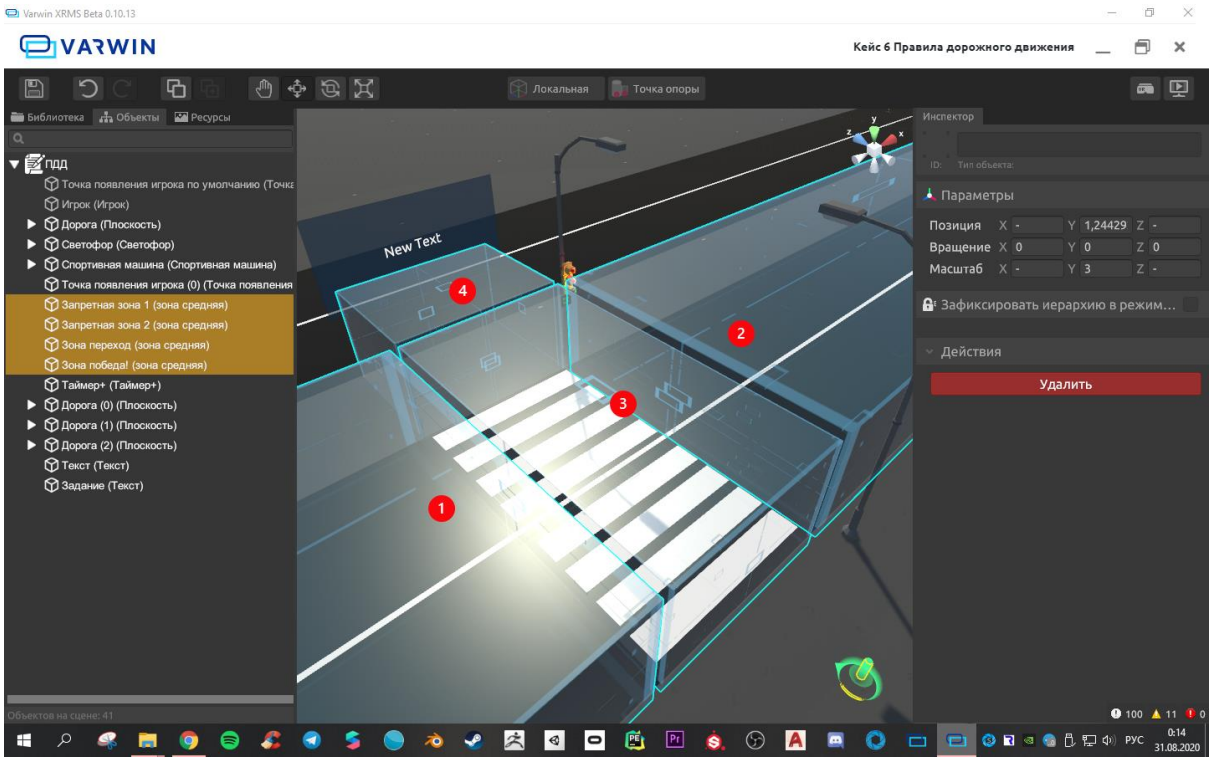
Добавляем на сцену объект “Спортивная машина” из пакета “Мегаполис”, эта машина будет ездить по нашей дороге и нужно будет вовремя перейти дорогу по пешеходному переходу и не попасть под дорогу.

Чтобы машина двигалась нам нужно на сцене создать четыре объекта “Точка” (1,2,3,4) по которым она будет передвигаться. Далее в редакторе логике Blockly мы создадим список точек, который преобразуется в маршрут для машины.



Расставляем вспомогательные объекты “Зона”

Для ограничения передвижений игрока нам понадобится создать четыре объекта типа “Зона” и расставить их в нужных местах. (1) и (2) - это запретные зоны, заходя в них игрок должен перемещаться обратно на тротуар. (3) - это зона перехода, в ней будут определяться столкновения с машиной и игрок будет допускаться в неё только на зеленый свет. (4) - это зона победы, в ней игрока будут поздравлять с победой, если он перешёл дорогу правильно.



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое “родительский” и “дочерний” объект?
2. Какие существуют источники освещения и какие у них отличия?
3. Каким образом реализовать пешеходный переход быстрее всего используя примитивные объекты?

Занятие 5.2 Функции

Цель:

Познакомиться с понятием “функция”, узнать что это такое, для чего предназначено и как использовать.

Задачи:

- Сформировать понимание определения “функция” в программировании
- Узнать какие типы функций существуют в Varwin
- Усвоить на практике как в Varwin используются функции

Давайте познакомимся с новым термином - это функция.

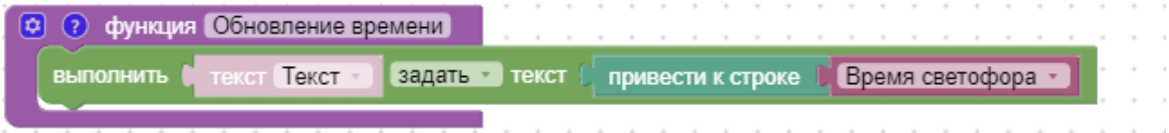
Определение: Функция в программировании, или подпрограмма — *фрагмент программного* кода, к которому можно обратиться из другого места программы.

Литература для подготовки к занятию:

1. Функция (программирование) — Википедия
2. Функции в программировании • Информатика, Практика программирования
3. Понятие функции в программировании

Непосредственно в Varwin существуют 2 типа функций: с возвратом переменной (2) и без возврата (1). Выглядит в Blockly это вот так:

В данном проекте мы будем использовать лишь одну небольшую функцию без возврата переменной, а просто выполняющую действие, чтобы познакомиться с самим блоком. Она будет обновлять время на текстовой панели под светофором. Назовём её “Обновление времени”.



Примечание: *в данной функции используется всего один логический блок, но на самом деле их может быть в функции достаточно много. В следующих кейсах мы будем использовать расширенные функции и вы поймете это на конкретных примерах.*

Используя опыт предыдущих занятий и сформировавшееся понимание возможного использования блоков в Blockly попробуйте реализовать **несколько простых функций**, которые будут состоять из 2 и более блоков внутри.

Обязательно обратите внимание на то, чтобы обучающиеся правильно называли сами функции. Объясните насколько это важно, также как и с именованием переменных.

Например, задайте последовательное перемещение объекта по локации с помощью одной функции.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие 2 типа функций существуют в Varwin?
2. Что такое функции и в каких случаях их удобно использовать?
3. Сколько блоков может быть в одной функции?

Занятие 5.3 Создание логики светофора

Цель:

Разработать полноценную логику образовательного проекта "Правила дорожного движения" согласно техническому заданию.

Задачи:

- Научиться применять функцию созданную на прошлом занятии
- Познакомиться с объектом таймер и его базовыми логическими конструкциями
- Закрепить навык работы с вкладками редактора логики
- Усвоить навыки тестирования своих проектов на баги/ошибки
- Закрепить навыки использования свойств объекта в редакторе логики
- Усвоить навык использования объекта “Зона”
- Научиться работать с освещением в редакторе логики
- Познакомиться с блоком список и создать простейший список

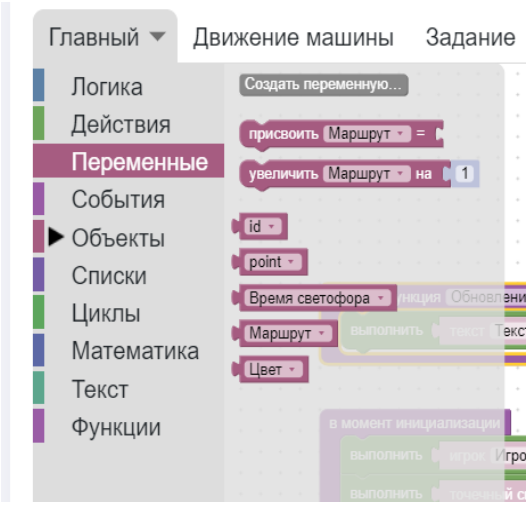
Методические материалы для подготовки к занятию:

Пошаговая инструкция по разработке кейсов #6.

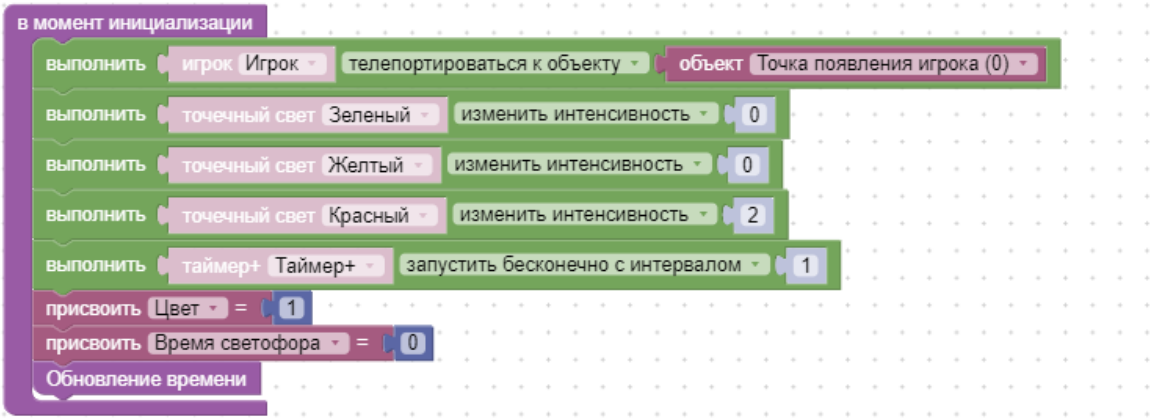
Собираем и настраиваем логику работы светофора

Первое что нам нужно сделать для успешной реализации логики данного кейса это создать три переменных, которые будут использоваться для подсчета

времени на светофоре, запоминания маршрута движения машины и изменении цвета светофора.

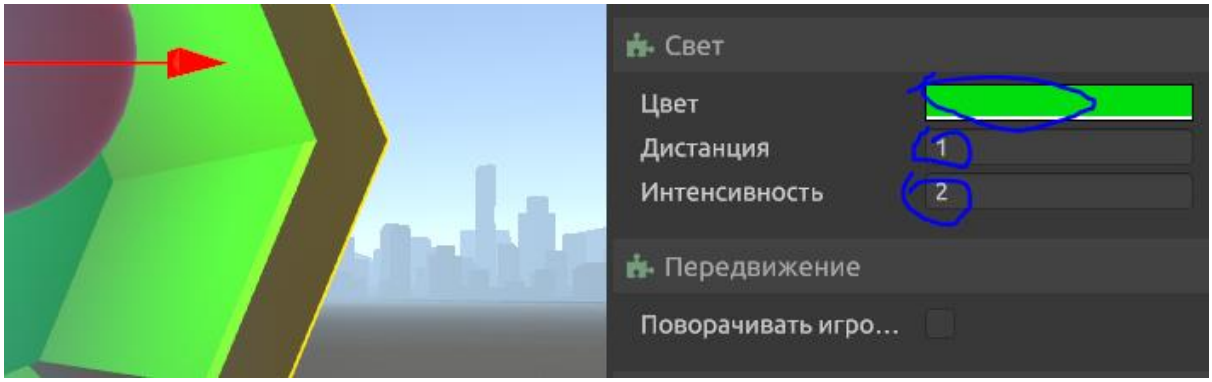


Далее нам понадобится уже знакомое событие - “В момент инициализации”. Поместим внутрь него следующие блоки. Здесь мы используем нашу функцию, которую мы создали на прошлом занятии - “Обновление времени”.



- Телепортируемся в точке появления игрока, чтобы всегда начинать игру в правильной позиции.
- Задаем увеличенную интенсивность света для красного точечного света. Да-да, мы можем изменять параметры точечного света и через редактор логики Blockly.

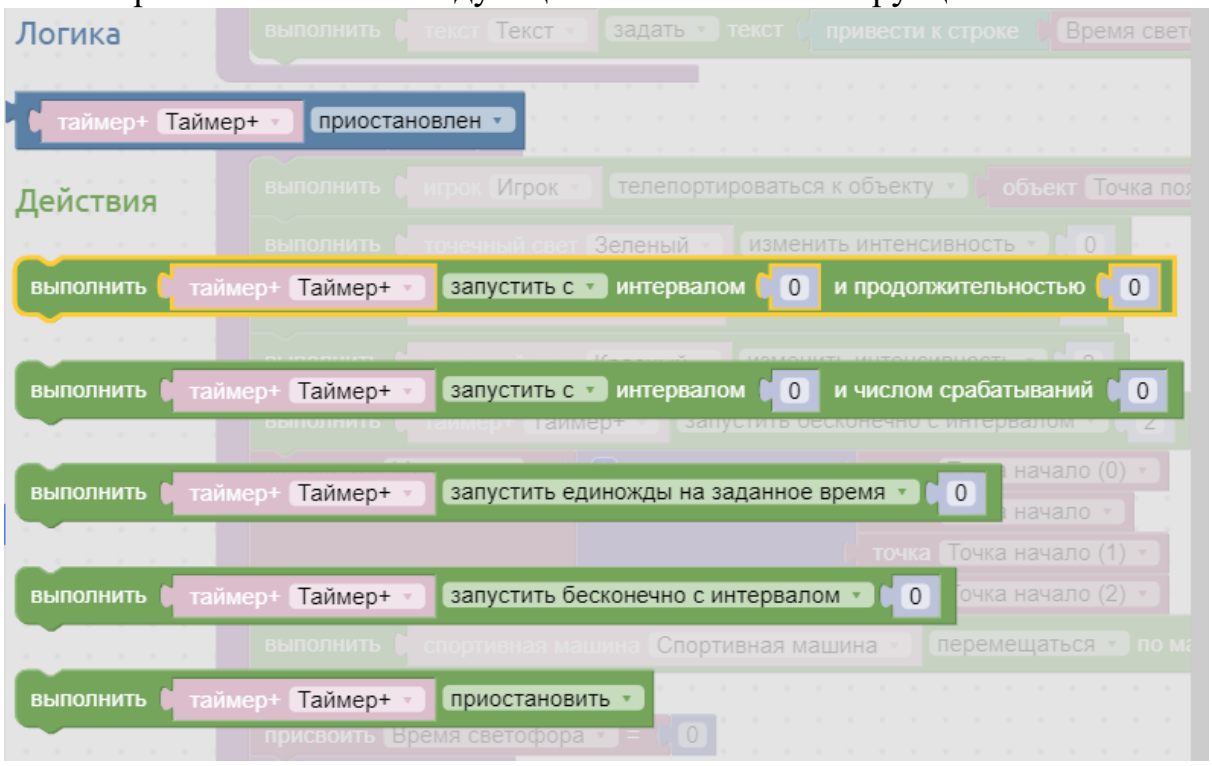
Также, давайте сразу рассмотрим свойства объекта “Точечный свет”. В нём можно изменять цвет света, дистанцию на которую он будет светить и интенсивность света.

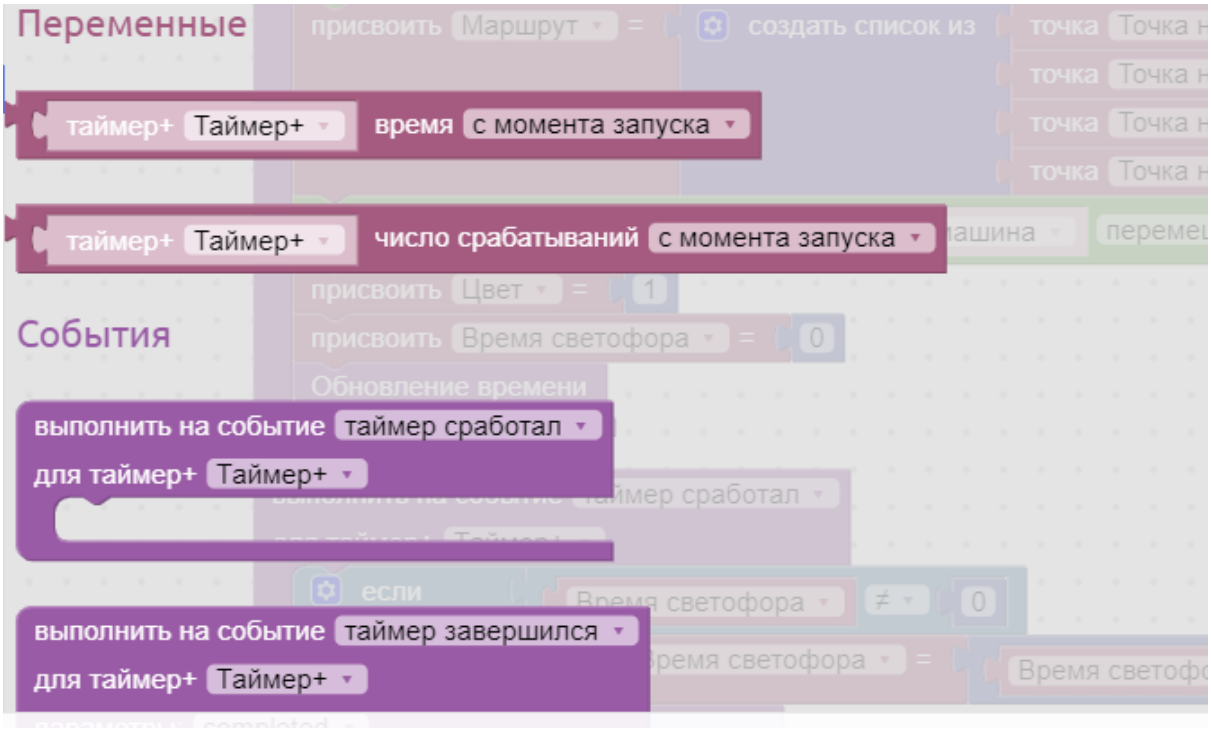


- Важный пункт нашей логической конструкции это таймер. Он должен работать бесконечно с интервалом 1 секунда, чтобы следующие блоки могли функционировать в нужной логике.

Определение: Таймер (от англ. Timer) *— в информатике средство обеспечения задержек и измерения времени средствами компьютера.*

У таймера в Varwin есть следующие логические конструкции:

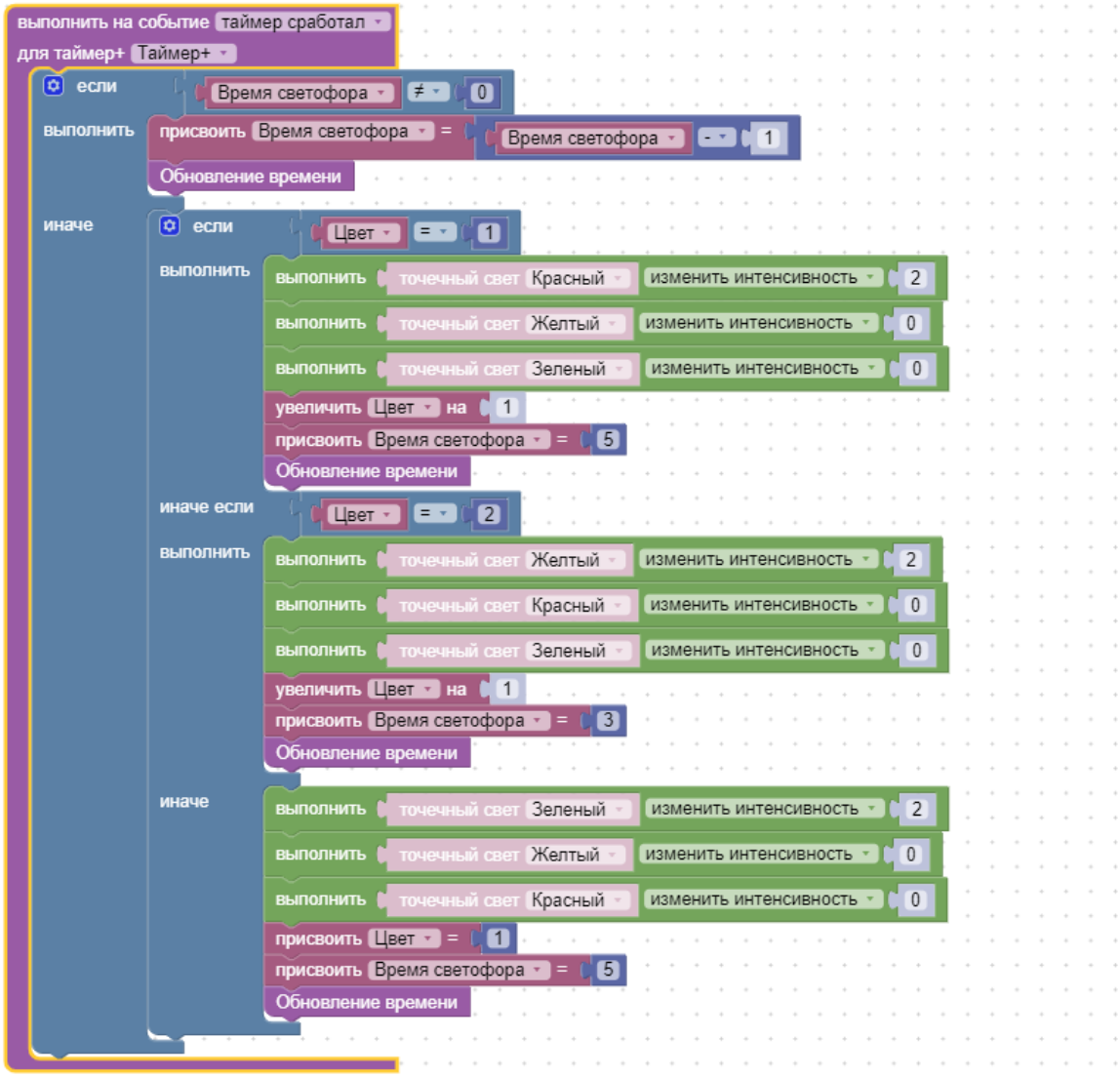




- Не забудьте присвоить значения переменным, которые мы создали ранее.
- И последнее, используем нашу функцию “Обновление времени” для вывода времени на светофор.

Далее, нам понадобится событие объекта “Таймер” - таймер сработал.

Внутри него мы реализуем логику переключения цвета светофора через определенное время, используя условные конструкции. Переменная время светофора будет отвечать за то, сколько будет светить определенный цвет. Зеленый и красный свет будут длится 5 секунд, желтый 3 секунды.



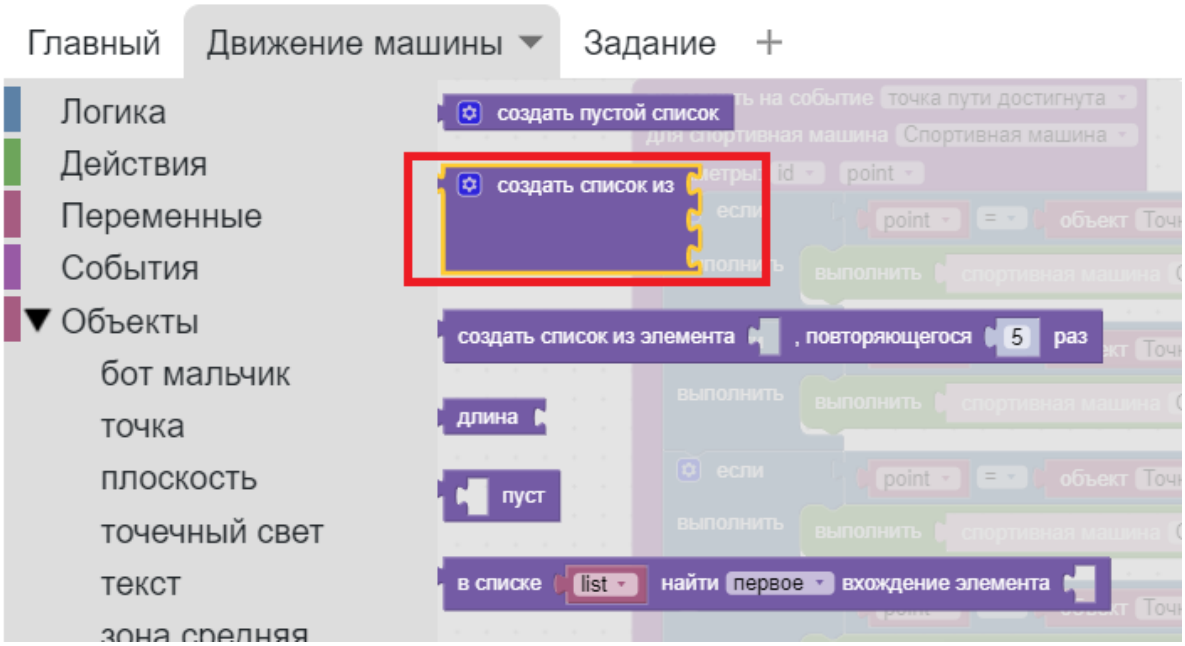
Чтобы переключать цвета светофора нам понадобится переменная “Цвет”. Каждый цвет будет обозначен своей цифрой. Красный - 1, Желтый - 2, Зеленый - 3.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

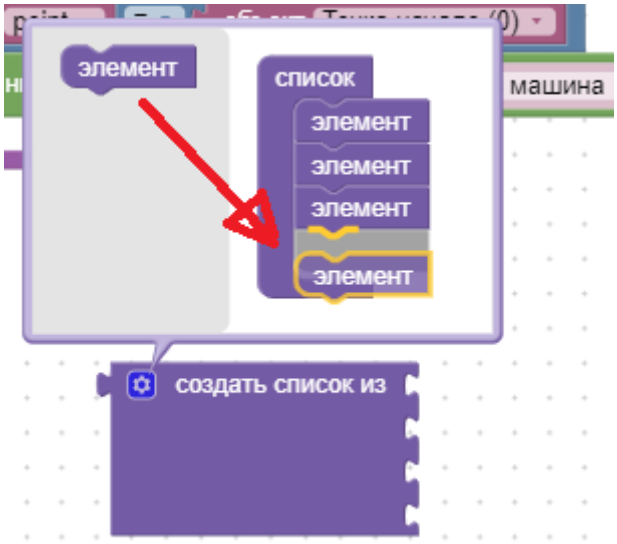
Логика движения машины по маршруту

Для того чтобы заставить нашу “Спортивную машину” двигаться нам необходимо познакомиться с новым типом данных - это список (более подробно мы рассмотрим его в следующем кейсе).

Стандартный список в Varwin создается из трех элементов, нам же нужно четыре элемента в списке. Для того, создаем список с тремя элементами:

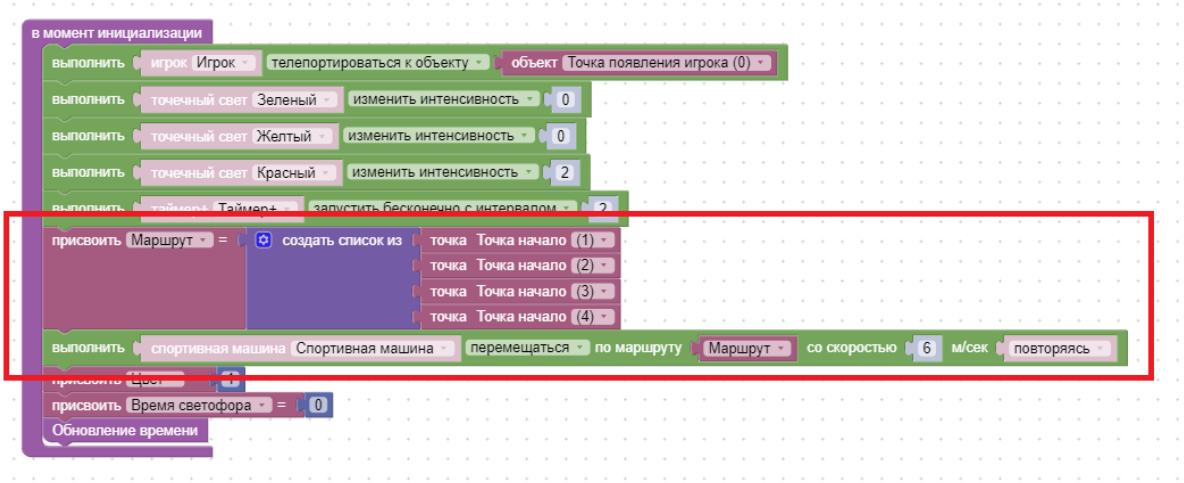


Далее, нажимаем на шестеренку и добавляем еще один элемент в список:



Теперь располагаем точки, которые мы создали для маршрута машины, в списке и присваиваем переменной “Маршрут” этот список.

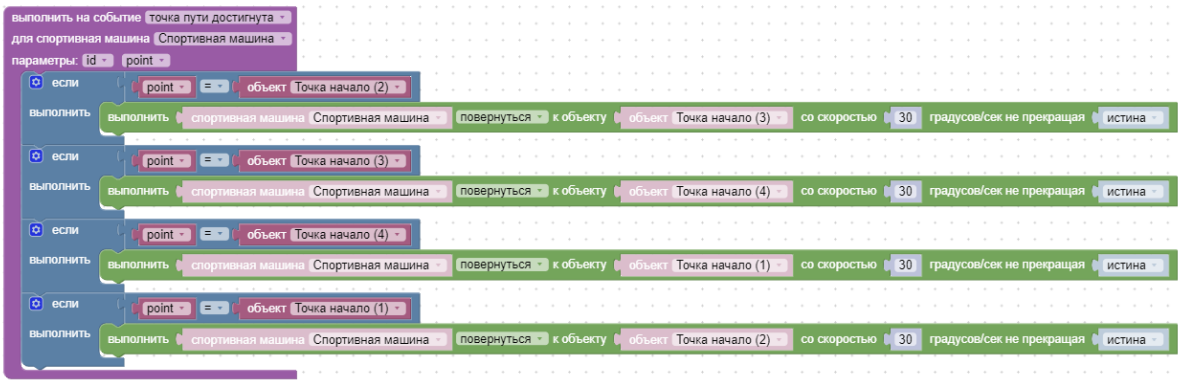
Также, задаем машине действие передвигаться по нашему новому маршруту.



Как вы заметили, у действия “Перемещение по маршруту” есть несколько свойств, давайте рассмотрим их подробнее.

Повторяясь	**Туда-сюда**	**Один раз**
Это означает, что машина будет двигаться по маршруту бесконечно зациклено, если нет других условий.	В этом случае машина проедет по маршруту в одну сторону и вернется в первоначальную позицию.	Если выбрано свойство “Один раз”, то машина проедет по маршруту один раз и останется в конечной точке.

Теперь нам только осталось сделать логику поворота машины в нужную сторону при преодолении поворотов на нашем маршруте. *Это будет самостоятельной работой.*
Финальная логика должна выглядеть так, создайте её в отдельной вкладке редактора логики:



Собираем логику задания

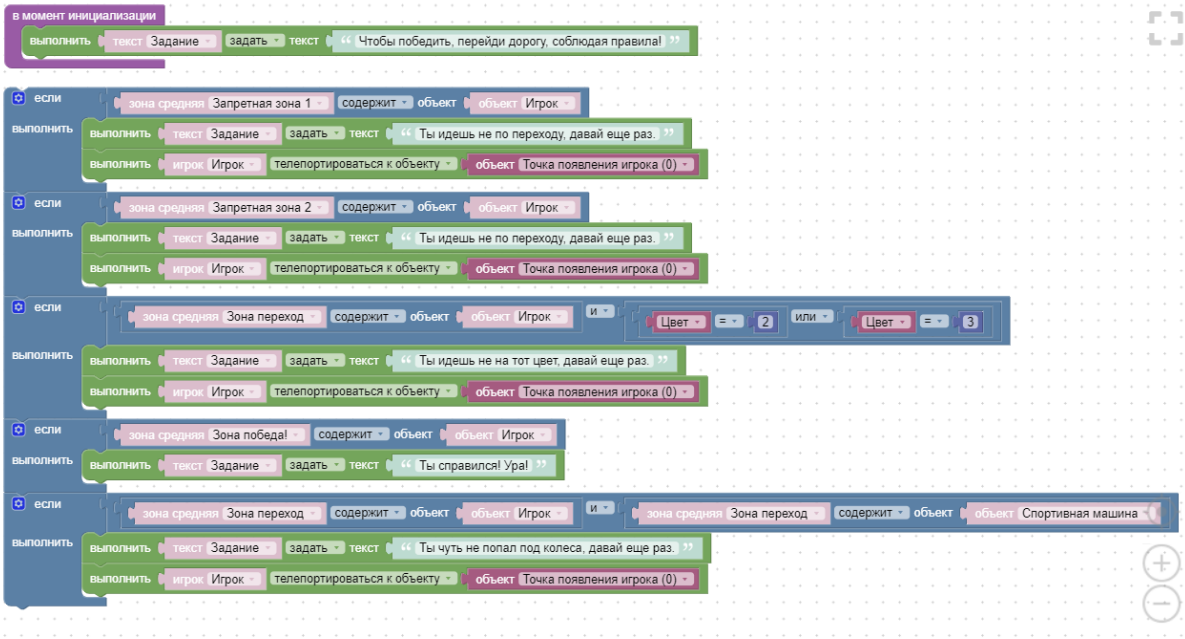
Последнее что нужно сделать в текущем кейсе, это собрать логику задания. Помните мы ставили вспомогательные объекты “Зона”? Ага, теперь нам нужно настроить их.

Давайте подумаем что нам нужно выводить на текстовую панель при вхождении в определенную зону и задать действия.

В момент инициализации задаём первичный текст задания.

При попадании в запретную зону игрок должен телепортироваться в “Точка появления игрока”.

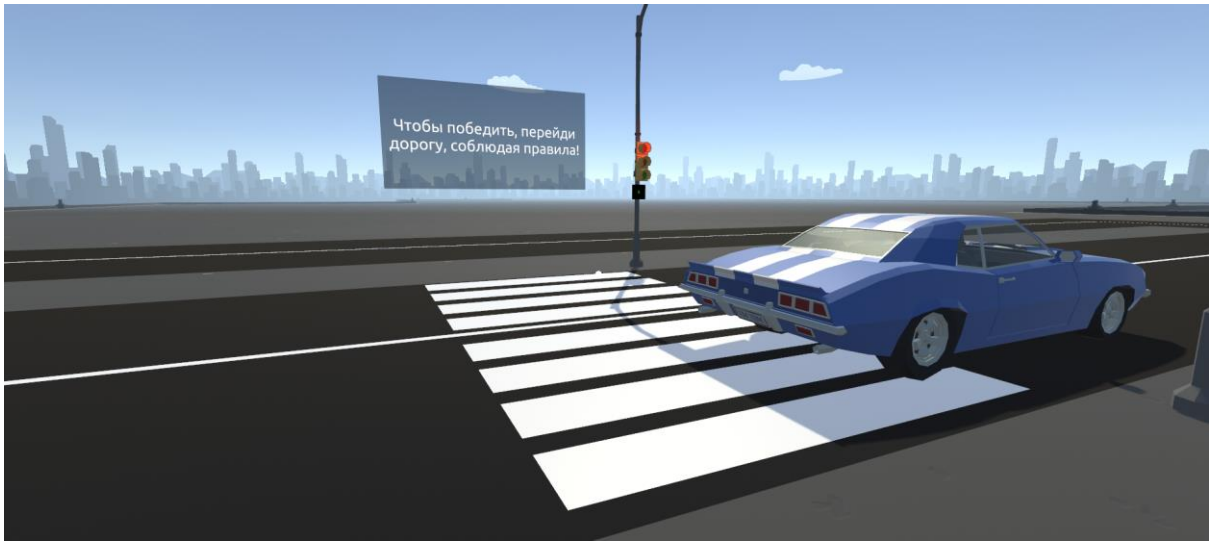
Остальную логику Вам необходимо реализовать самостоятельно. Также, создайте её в отдельной вкладке редактора логики. Финальный вид логики должен быть примерно таким:



Обязательно протестируйте собранное приложение на работоспособность, если что-то не работает, то ищите баги и исправляйте их. Если что-то не так попробуйте свериться с нашим маршрутом разработки данного кейса.

Если у вас всё заработало - поздравляем! Мы закончили сборку еще одного увлекательного кейса и можем двигаться дальше. Мы изучили много нового материала. Создали свой первый список, познакомились с функциями, узнали что такое точечный свет и как его настраивать. Закрепили навыки работы с зонами, примитивами и уже изученными логическими блоками.

Финальный вариант кейса выглядит подобным образом:



Рефлексия. Получилась ли у Вас локация соответствующая техническому заданию? Что было реализовать сложнее всего? Сложно было работать с условиями и зонами? У всех получилось? Переменные в совокупности с условиями тоже выглядят сложными конструкциями, у всех появилось понимание как у нас работает светофор?

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие есть свойства у действия передвижение по маршруту? Чем они отличаются?
2. Что такое таймер и для чего его можно использовать?
3. В чем разница между событиями «таймер сработал» и «таймер завершился»?
4. Какие свойства есть у объекта точечный свет и как их можно использовать в Blockly?
5. Как работают множественные условия и какие логические операторы в условиях существуют?

Занятие 5.4 Расширение проекта ПДД

Цель:

Усвоение навыков, полученных в ходе практических занятий. Развитие (апгрейд) проекта “Правила дорожного движения” по собственному техническому заданию.

Задачи:

- Усвоить навыки работы с иерархией в редакторе Varwin
- Закрепить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Закрепить навык работы с логикой базовых блоков в Blockly
- Закрепить навык работы с desktop - редактором XRMS Varwin

- Повысить навыки рисования скетчей/ планов перемещения по виртуальному пространству
- Закрепить навыки работы с локацией и размещением объектов на сцене
- Закрепить навыки создания и использования дизайна интерфейсов, основанном на удобстве для конечного пользователя
- Усвоить навыки работы с функциями в Varwin

Формат: Самостоятельная работа, решение кейсового задания.

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

На этом этапе нужно усвоить навык формирования и фиксации технического задания.

Шаблон технического задания:

1. Тематика/направленность проекта;
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы или же свои объекты);
3. Основной функционал приложения;
4. Дополнительные возможности приложения;
5. План-схема сцен, которые будут реализованы;
6. Каким образом будет реализован, удобный для пользователя, UX/UI - дизайн;
7. Какую полезную функцию для общества несет проект, или будет нести при дальнейшей его доработке.

Рекомендация: если есть возможность, то посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом обучающимися.

Кейс:

Необходимо расширить проект “Правила дорожного движения”. Сейчас вам предоставляется творческая свобода и Вам решать как расширять этот проект. Вы можете создать дополнительные переходы, создать более сложный маршрут движения или добавить машину полиции. Самое главное соблюдайте обязательные условия.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта
2. Нарисовать план расположения объектов на сцене
3. Зафиксировать дополнительные функции, которые будут реализованы в проекте.
4. Используйте дополнительную функцию с минимум 3 действиями внутри функции.
5. Реализовать дополнительный светофор для машины и механику остановки машины перед светофором.

6 Списки

Занятие 6.1 Списки

Цель:

Познакомиться с понятием “список”, узнать что это такое, для чего предназначено и как использовать в своих проектах на Varwin.

Задачи:

- Сформировать понимание определения “список” в программировании
- Узнать какие списки существуют в Varwin
- Познакомиться с логическими блоками списков в Blockly
- Рассмотреть ситуации в которых можно использовать списки

Методические материалы для подготовки к занятию:

Литература для подготовки к занятию:

Список в информатике

Что можно делать со списками в программировании?

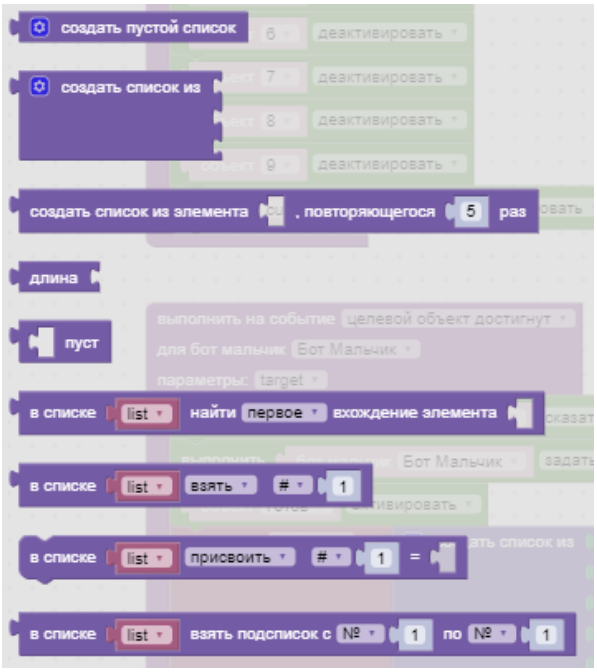
Определение:

В информатике, спи́сок (англ. list) *— это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Экземпляр списка является компьютерной реализацией математического понятия конечной последовательности. Экземпляры значений, находящихся в списке, называются элементами списка (англ. item, entry либо element); если значение встречается несколько раз, каждое вхождение считается отдельным элементом.*

Простыми словами Список Varwin - это переменная, которая хранит в себе множество однотипных значений, каждое из этих значений имеет порядковый номер (индекс).

Сейчас нам нужно изучить основные блоки списков, для того чтобы понять как они работают и иметь представление в каких проектах их использовать.

Основные логические блоки у списков:



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое список в информатике?
2. В каких случаях удобно использовать списки?
3. Что такое пустой список и для чего его используют?

Занятие 6.2 Бот Varwin

Цель:

Узнать что такое “Бот” в Varwin. Собрать первую часть локации нового проекта.

Задачи:

- Узнать кто такие боты и для чего их можно использовать
- Познакомиться с логическими блоками ботов в Blockly
- Закрепить навык расстановки объектов на локации
- Научиться настраивать логику бота
- Закрепить навык тестирования своих проектов

Методические материалы для подготовки к занятию:

Первое что нам нужно изучить на этом занятии это объект типа “Бот” и какой функционал он в себе несёт.

Ссылка на актуальную документацию Varwin:

Varwin Bot — Документация Varwin 0.7.0 Beta

Нам нужно усвоить информацию о том как можно управлять перемещением ботов и как научить их разговаривать.

Основные логические блоки ботов очень хорошо структурированы в нашей документации. Можно ознакомиться с ней по ссылке выше.

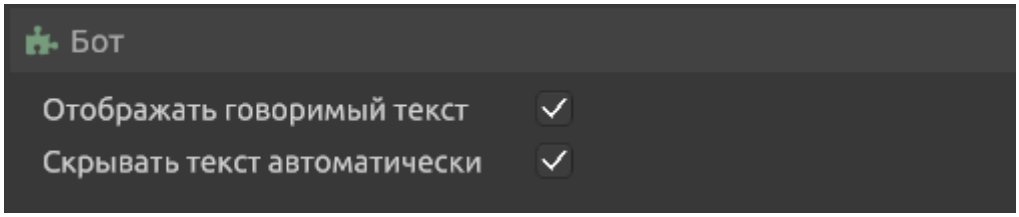
Давайте разберем какие есть уникальные блоки Бота, которые нам пригодятся в текущем проекте.Таблицу ниже вы можете сохранить и использовать в качестве справки.

***Примечание**: здесь мы отметим только важные блоки, которые не встречаются в стандартной логике, вы можете самостоятельно поэкспериментировать, чтобы узнать за что отвечают остальные блоки объекта "Бот".*

№	Логический блок	Описание блока
	Действия	
1	Запустить анимацию **X**	Запускает анимацию бота, при ее наличии. У стандартного бота нет анимаций, но вы можете встретить расширенных ботов у которых могут быть, например, анимации эмоций (грусть/радость и др.).
2	Сказать **<Текст>**	Активирует облачко текста над головой бота.
3	Сказать **<Заголовок> <Текст>**	В этом случае можно задать заголовок. Как правило заголовок используется, чтобы задать имя или статус персонажа.
4	Перестать говорить	Скрывает облачко текста.
5	Задать отображение говоримого текста **Истина/Ложь**	От этого будет зависеть будет ли появляться облачко текста над персонажем или нет. Также можно задать это свойство в инспекторе.
6	Задать тип скрывания говоримого текста **Никогда/Автоматически**	Определяет будет ли текстовое облако скрыться через некоторое время после произнесения текста или будет висеть всегда или до наступления определенного события.
	События	
1	Фраза произнесена	Запускает для выполнения алгоритм действий, помещенный внутри события при выполнении условия, что Бот только что произнес **любую** фразу.

Также у ботов есть дополнительные свойства в редакторах. Можно отображать/скрывать говоримый текст и скрывать текст автоматически.

1. **Отображать говоримый текст** - бот будет общаться с нами посредством текстовых облаков над головой. Это свойство отвечает за то, будут ли они отображаться. В нашем случае нам это свойство требуется активировать.
2. **Скрывать текст автоматически** - а вот у этого свойства флажок можно снять, чтобы облачка текста не пропадали со временем, а постоянно висели над головой бота.



Теперь давайте поговорим о задании на этот модуль.

Техническое задание:

Необходимо разработать приложение на Varwin для изучения простых английских слов: числительных и предметов интерьера.

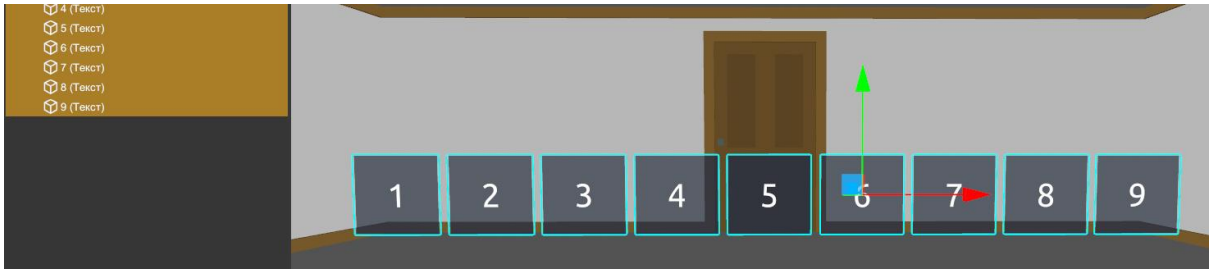
Специальные требования:

1. Приложение должно состоять из двух частей (сцен): на первой сцене мы проверяем знание числительных, а на второй знание объектов интерьера.
2. Задания должен давать виртуальный помощник (бот).
3. Должна быть реализована система оценки в баллах.
4. На второй локации требуется реализовать ограничение по времени выполнения задания (30 секунд), и времени на ответ (10 секунд).
5. Задания должны выдаваться в случайном порядке, чтобы при каждом новом запуске приложение работало по новому алгоритму.

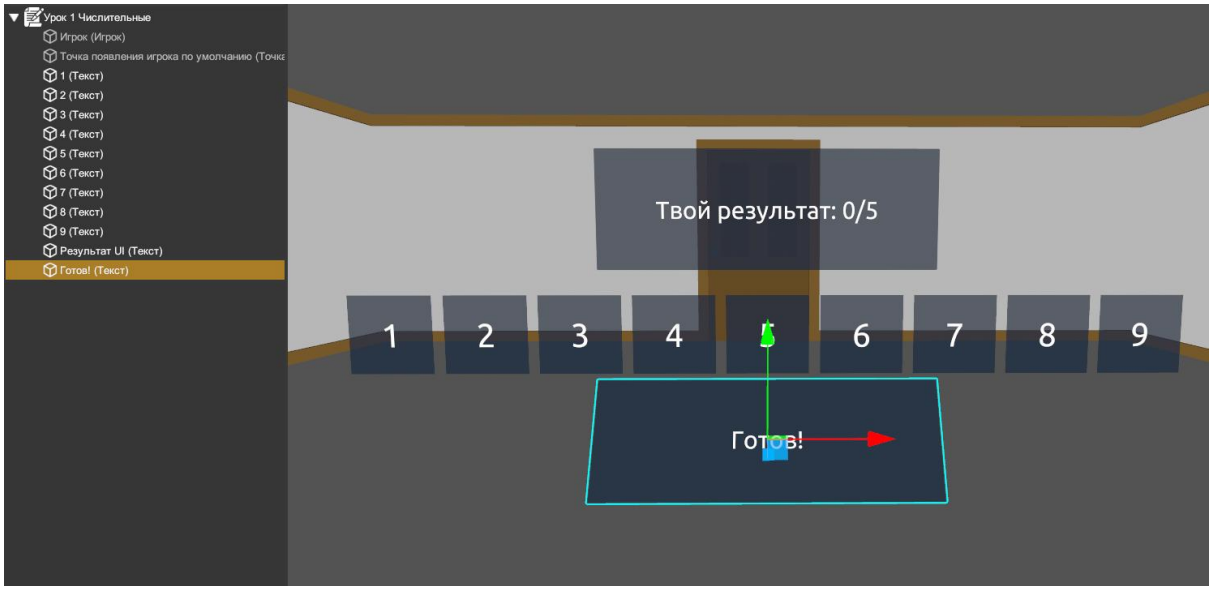
Сборка сцены

Создадим новый проект и добавим к нему сцену “Гостиная”. Давайте разместим на сцене необходимый UI и объекты.

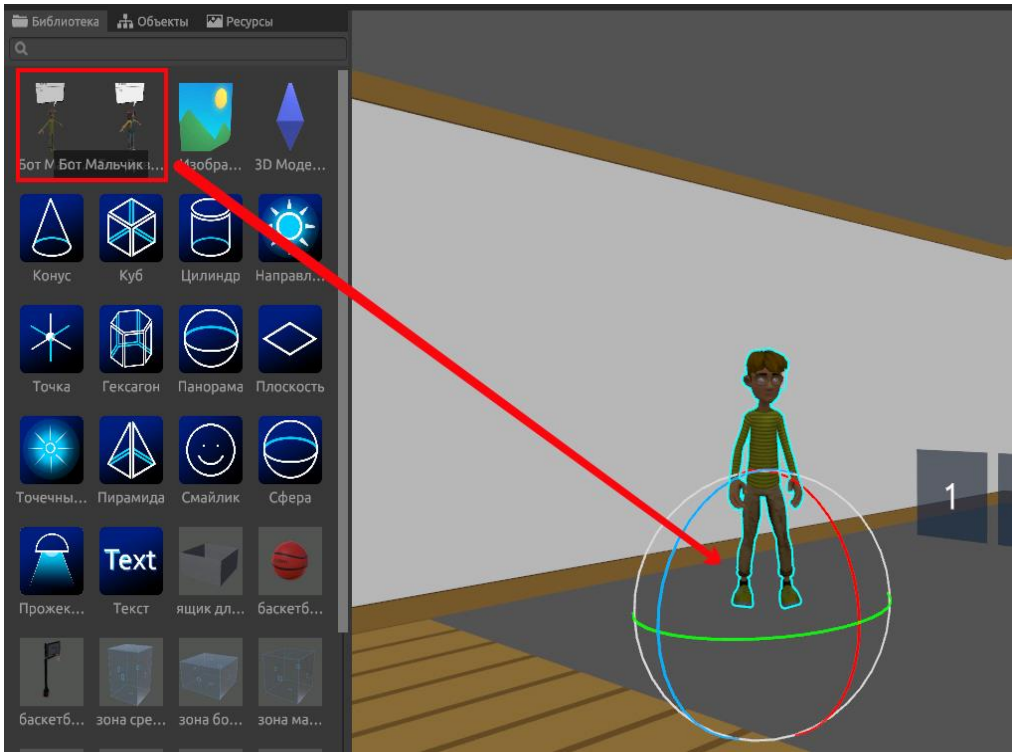
Для начала расположите на сцене UI с числительными от 1 до 9:



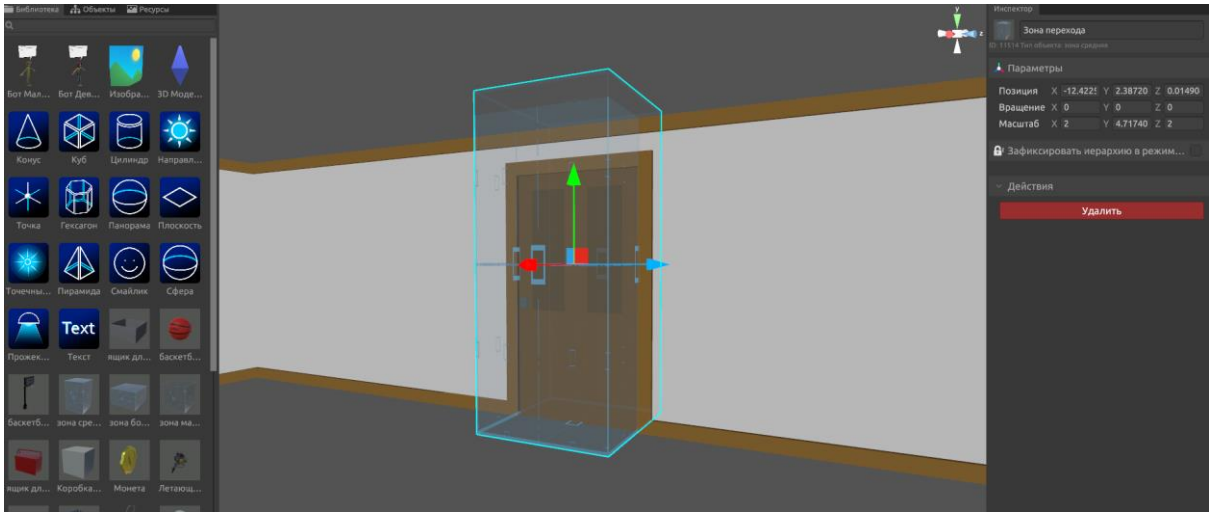
Выше расположим UI для отображения результата и UI, которое будет стартовать задание при его использовании.



Также разместим на локацию Бота (объект *бот Мальчик*) и расположим его подальше от игрока, чтобы в начале сценария он к нам подбежал.



И последнее, что нам необходимо сделать - это расположить зону в районе двери, чтобы в будущем настроить логику перехода на другую сцену.



Все необходимые объекты расположены на сцене. Теперь, на следующем занятии, мы можем перейти к созданию логики.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Кто такой “Бот” и для чего он нужен?
2. Какие два основных комплекса логических блоков есть у бота?

Занятие 6.3 Сборка логики первого задания

Цель:

Разработать первую сцену образовательного проекта "Урок английского".

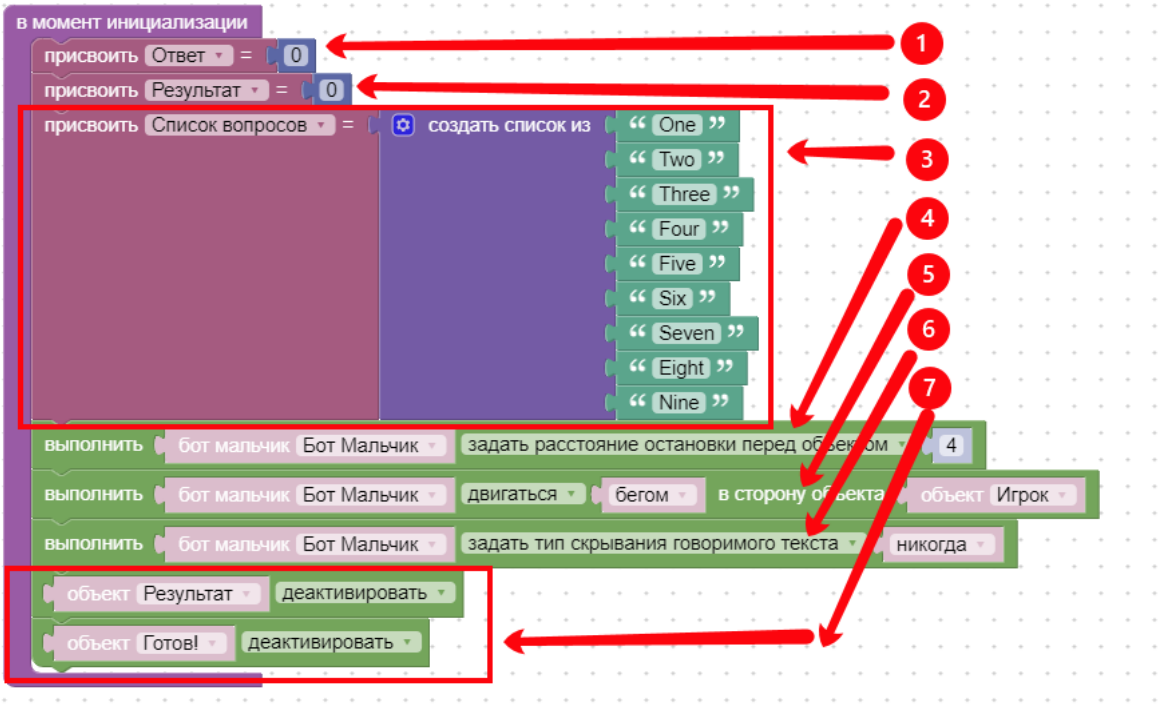
Задачи:

- Закрепить знания о таком понятии как список
- Закрепить навыки работы с логическими блоками ботов в Blockly
- Закрепить навыки настройки логики ботов
- Закрепить навыки тестирования своих проектов
- Создать необходимые функции для реализации проекта
- Реализовать логику задания первой сцены
- Научиться работать с активацией/деактивацией объектов в Blockly
- Сформировать понимание как использовать функции для создания простейших циклов

В момент инициализации

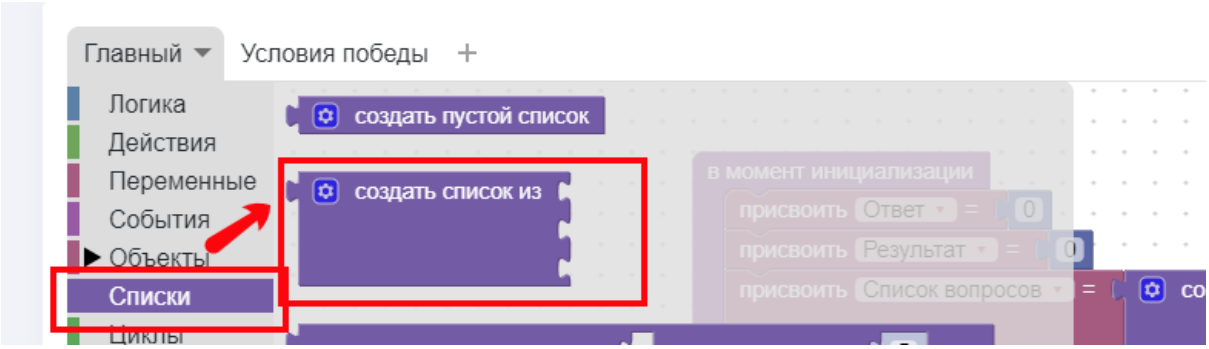
Для начала давайте определимся с логикой, которая будет активироваться в момент инициализации проекта, т.е. в самом начале работы приложения при запуске.

На картинке ниже мы видим событие “В момент инициализации”, теперь давайте пошагово разберемся, что значит каждый блок в теле этого события:

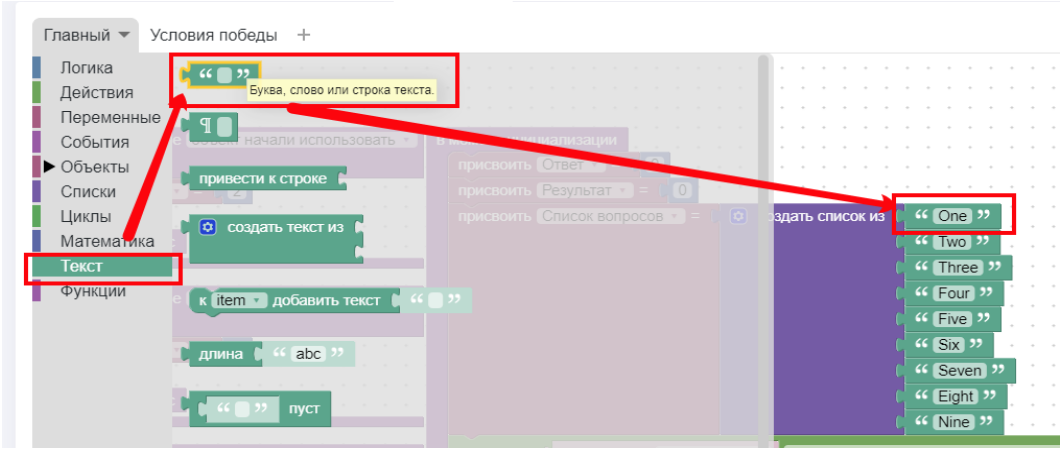


1. Во первых нам необходимо объявить переменные *Ответ* (1) и *Результат* (2). *Ответ* будет соответствовать нашему ответу на вопрос бота. При правильном *Ответе Результат* будет увеличиваться на один. Пока положим в них значение “0”.

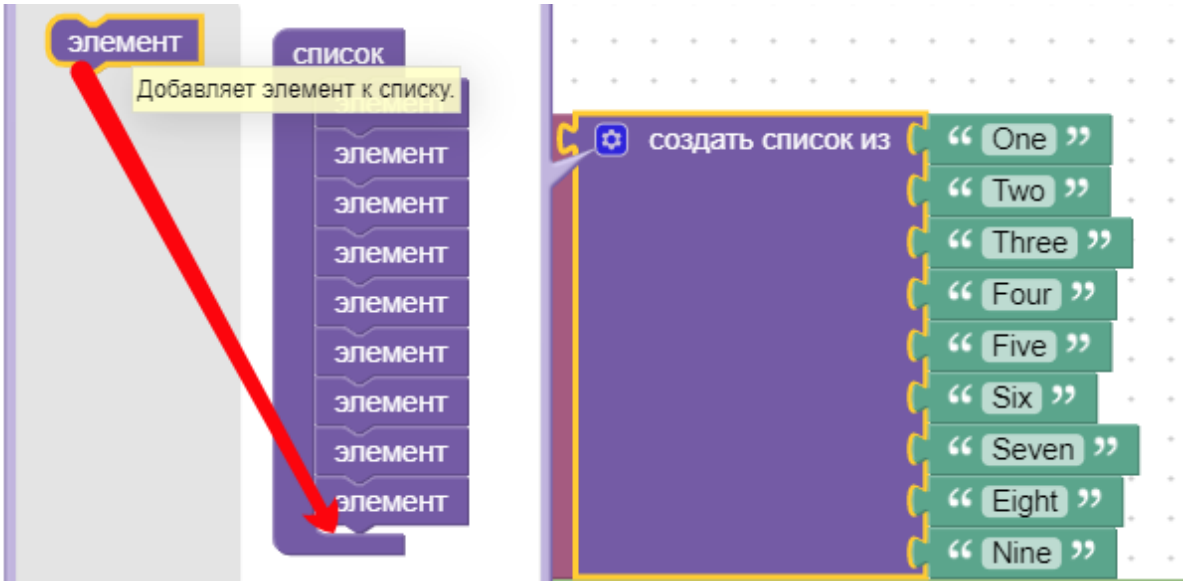
2. Далее мы создаем список вопросов (3).
   Чтобы создать список необходимо создать переменную (в нашем случае это переменная *Список вопросов*) и далее выбрать блок **“Создать список из”**.



И после следует добавить туда английские наименования числительных от 1 до 9.

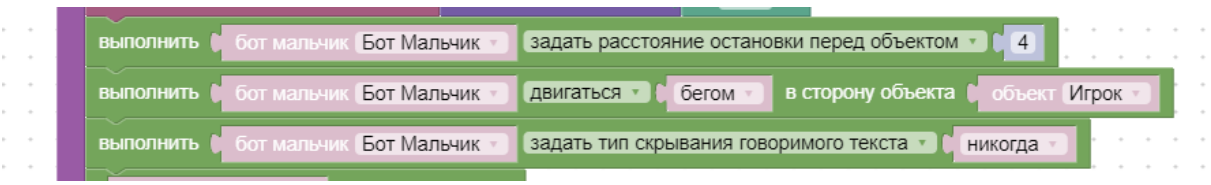


***Примечание**: чтобы расширить список дополнительными элементами необходимо нажать на "Шестеренку" и подставить вспомогательный блок "Элемент" в список как на картинке ниже.*



Далее Бот будет обращаться к этому списку, чтобы задавать нам вопросы.

3. Перейдем к блокам Бота.



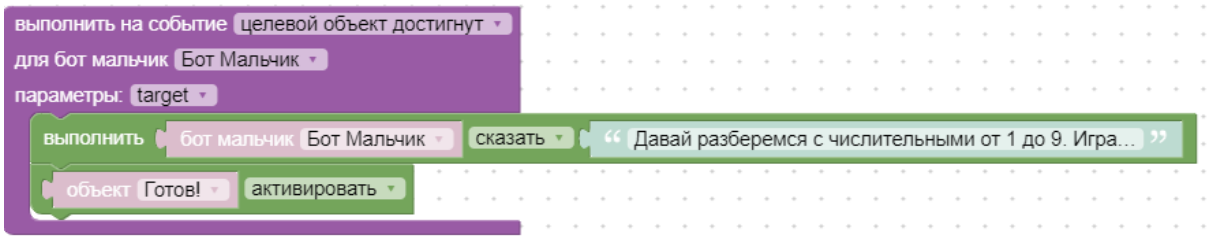
Разберем их на нашем примере.
Блок **"Задать расстояние остановки перед объектом"** (4) уже встречался нам в таблице стандартной логики, он присутствует и у Бота. Этот блок необходим, чтобы Бот остановился за 4 метра до игрока, иначе он будет пытаться пройти сквозь нас.

Блок **“Двигаться в сторону объекта X”** (5) несколько отличается от стандартного логического блока перемещения в сторону определенного объекта тем, что здесь мы можем задать анимацию перемещения *Шагом* или *Бегом*.
Блок **“Задать тип скрываемого текста”** (6) дублирует свойство объекта, которое можно установить в инспекторе, что мы и сделали ранее сняв флажок **“Скрывать текст автоматически”** в Desktop-редакторе. Этот блок здесь для наглядности.

4. Последнее действие в момент инициализации - нам необходимо деактивировать блоки UI *“Готов!”* и *“Результат”* (7), они будут появляться в момент, когда бот подбежит к нам и произнесет начальную фразу.

Начало задания и уникальные логические блоки объекта Бот.

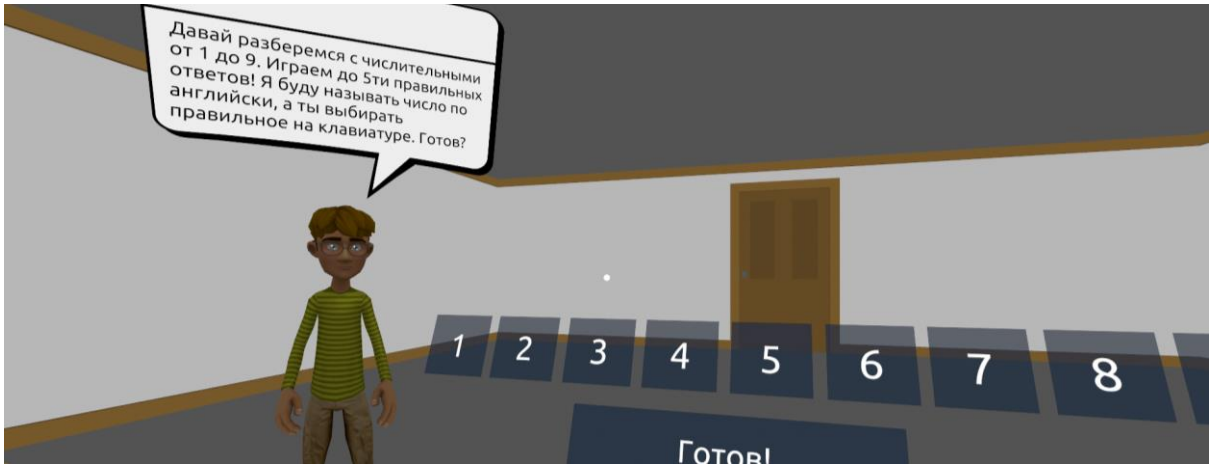
Итак, в момент инициализации Бот начинает двигаться в сторону объекта игрок.
В момент когда он достигает целевого объекта (при расстоянии остановки 4 метра перед целевым объектом), Бот должен произнести фразу с постановкой задачи перед Игроком, в нашем случае формулировка такая: “Давай разберемся с числительными от 1 до 9. Играем до 5ти правильных ответов! Я буду называть число по английски, а ты выбирать правильное на клавиатуре. Готов?”.



После этого мы активируем UI “*Готов*!”.

Блок “*Сказать*” является уникальным для Бота, он активирует облако текста над головой Бота с заданным текстом.

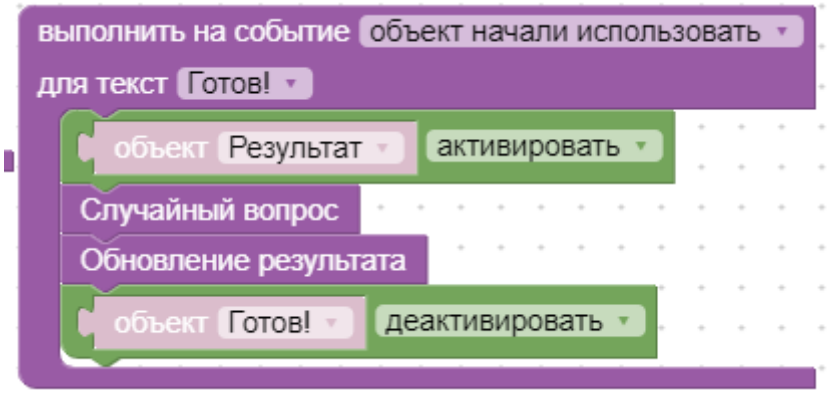
***Совет**: мы только что собрали базовую часть нашего приложения. Протестируйте, все ли работает так, как вы задумали. Если это действительно так, то вы увидите примерно такую картинку:*



Создание необходимых функций

Вы помните, что после того как Бот подбежал к нам и произнес фразу с постановкой задачи у нас должен активироваться UI **“Готов!”**.

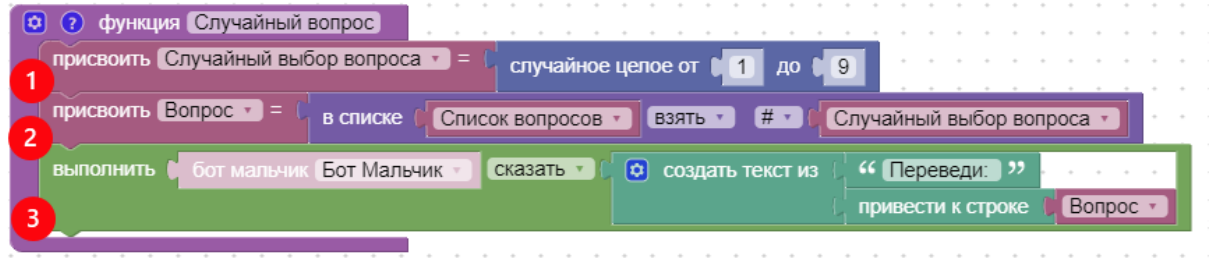
Далее нам необходимо собрать блок с логикой того, что происходит после использования этого UI:



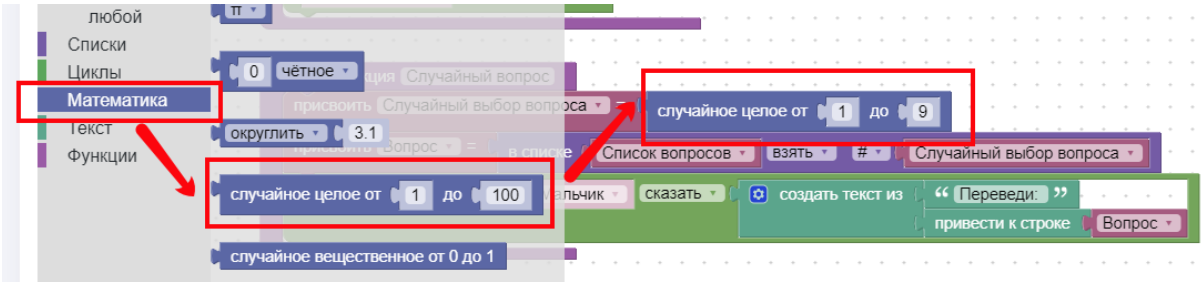
Как видите, здесь уже присутствуют функции “Случайный вопрос” и “Обновление результата”. Разберем их последовательно.

Функция Случайный вопрос

Данная функция выглядит так:

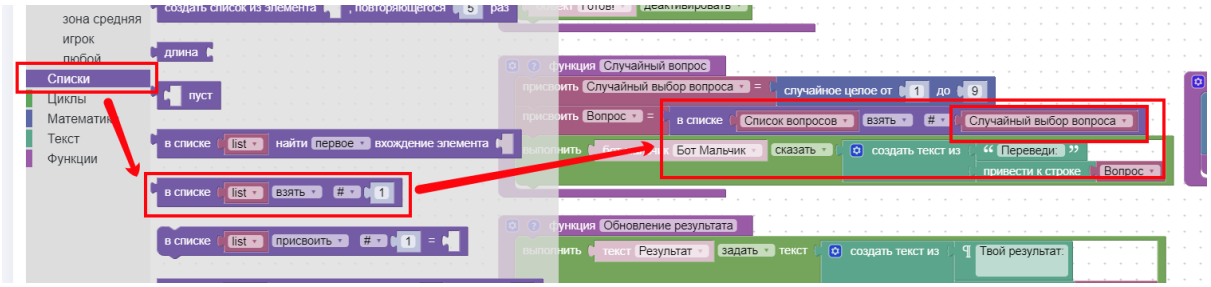


1. Для начала разберемся со случайными числами. Объявим переменную *Случайный выбор вопроса* и выберем блок в категории “Математика” **Случайное целое.** Установим диапазон от 1 до 9.



Этот блок будет генерировать случайное число от 1 до 9 каждый раз при активации этой функции для выполнения требования из ТЗ.

2. Далее нам необходимо присвоить новое значение переменной *Вопрос* (именно этот вопрос будет задавать Бот). Для этого в категории “Списки” выберем блок **В списке взять # (индекс) X.**

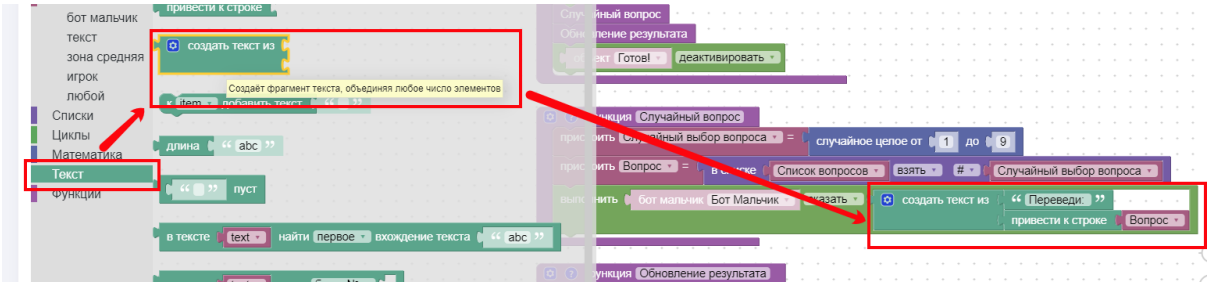


Наш список автоматически пронумерован от 1 до 9.
Подробно механика работает так: генератор случайных чисел из блока “**Случайное целое”** генерирует случайное число от 1 до 9, далее блок **“В списке взять # (индекс) X”** берет из списка текст с индексом, соответствующий этому случайному числу и присваивает этот текст в качестве значения переменной *Вопрос*. Далее этот вопрос задает игроку Бот.

3. Последний блок отвечает за фразу, которую произнесет Бот.
Здесь мы познакомимся с расширенным использованием блоков из категории “Текст”.

Текст можно не только вводить вручную, но и создавать из нескольких элементов, в том числе используя переменные.

В этом нам поможет блок из категории “Текст” **Создать текст из**.



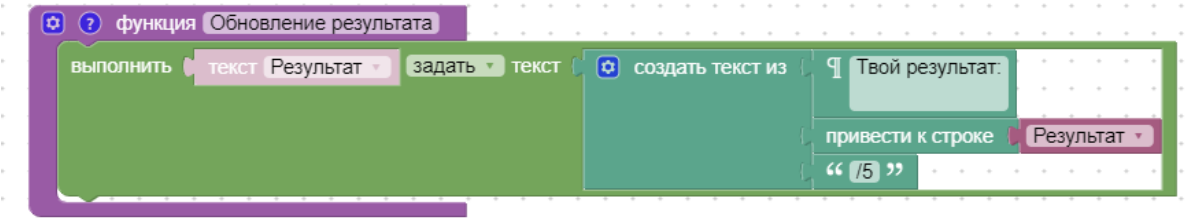
В нашем случае мы создаем текст из фразы “Переведи:” и второй элемент - это переменная Вопрос.

***Примечание**: при работе с текстом важно, чтобы все элементы были одного типа. Переменная Вопрос сейчас у нас числового типа, поэтому необходимо привести ее к текстовому типу, используя блок **“Привести к строке” из категории “Текст”**, иначе это может привести к логической ошибке. Обращайте на это внимание!*

Мы закончили работать с этой функцией, перейдем к следующей.

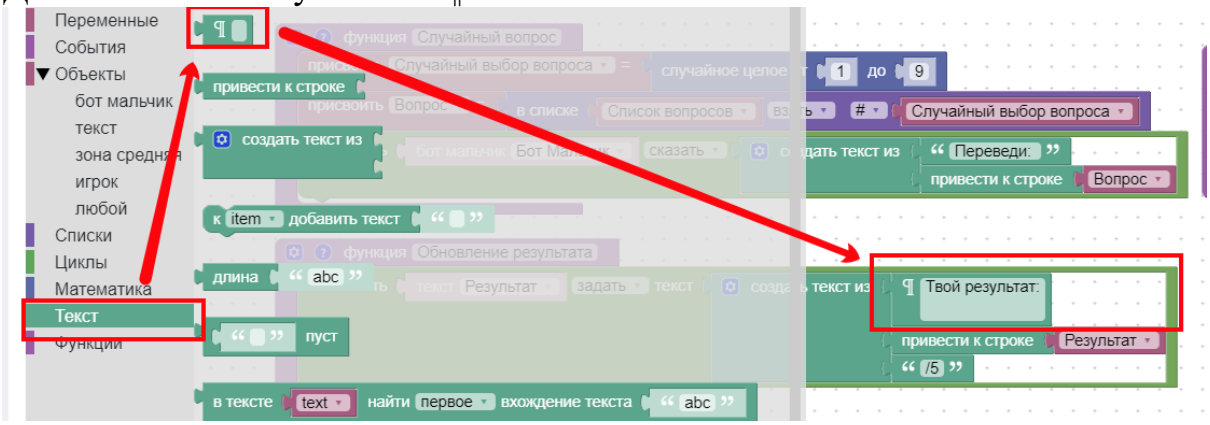
Функция Обновление результата

Эта функция отвечает за обновление информации на UI “Результат” и выглядит так:



Здесь используется уже знакомый нам блок “Создать текст из”, но здесь важно реализовать возможность переноса строки.

Для этого используем блок ¶:



При использовании блока “Создать текст из” он будет просто размещать элементы друг за другом в одной строке.

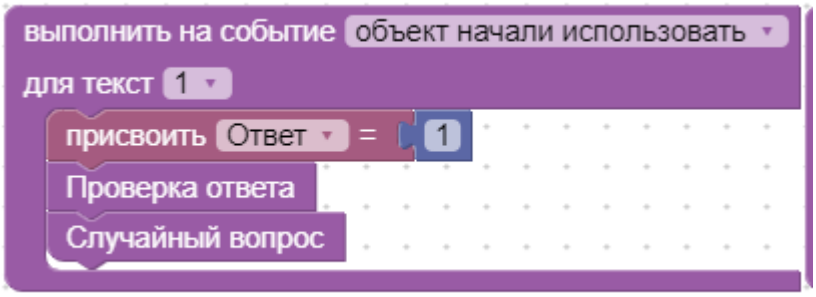
Блок ¶ отвечает за многострочный ввод. То есть, если мы напишем “Твой результат”, а после нажмем Enter и оставим пустую строку, то это будет восприниматься программой как перенос строки. Таким образом можно вводить текст, каждое предложение которого будет начинаться с новой строки.

Мы закончили работать с начальными функциями. Теперь нам необходимо настроить взаимодействие игрока с интерфейсом и проверку правильности ответа.

Настройка взаимодействия и Функции Проверка ответа

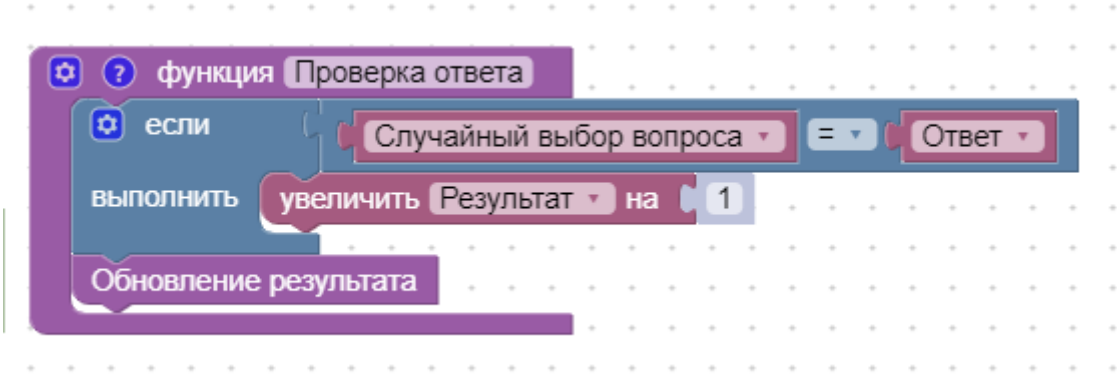
Итак, сейчас по логике у нас получается следующая картина: после того как мы получили задание и нажали на кнопку “Готов!”, бот задаст нам первый случайный вопрос, например *“Переведи: One”*. Далее нам необходимо нажать на кнопку *“1”*, чтобы дать верный ответ.

Давайте настроим эту логику. Для начала возьмем блок **“Объект начали использовать”**:



Вы видите, что при использовании UI “1” мы присваиваем переменной *Ответ* значение “1”, что совершенно логично, и активируется уже знакомая функция *Случайный вопрос,* которая заставляет бота задать следующее задание и еще незнакомая функция *Проверка ответа.* Давайте разберем последнюю.

Функция выглядит так:



Логика здесь следующая: мы помним, что первоначально генератор случайных чисел выбирает индекс из списка *Список вопросов*, соответственно, если наш *Ответ* равен этому индексу, то мы увеличиваем переменную *Результат* на 1 (инкрементируем) и активируем функцию Обновление результата, чтобы актуализировать информацию на UI “**Результат**”.

А после нашего ответа активируется функция *Случайный вопрос*, которая продолжает этот цикл.

Самостоятельная работа: соберите все события для всех UI с числами, чтобы у вас получилась следующая картина:

Вы еще раз можете убедиться для чего нужны функции - они выполняют повторяющиеся операции, что позволяет оптимизировать логику, ведь без функций нам бы пришлось описывать логику задавания случайного вопроса и проверки ответа в каждом из этих блоков выше.

Таким образом мы зацикливаем процесс задавания вопросов ботом с помощью этих функций, но до какого момента?

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое списки и в каких случаях их применяют?
2. Зачем создавать пустой список?
3. Какие логические блоки есть в категории «Список»?

Занятие 6.4 Добавление новой сцены в проекте

Цель:

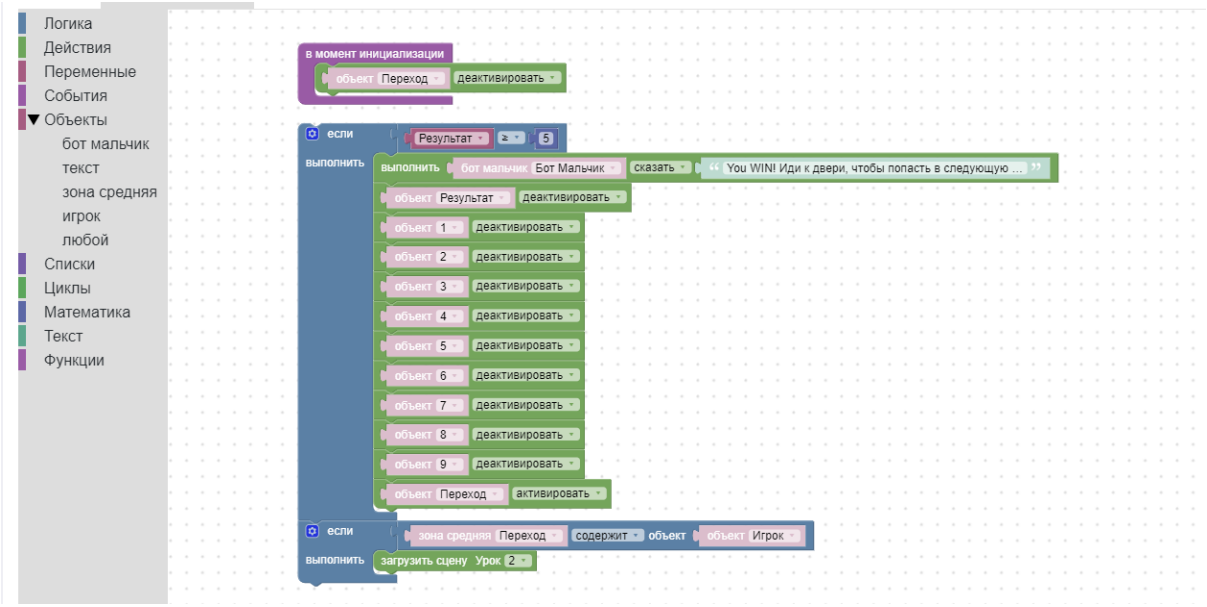
Разработать вторую сцену образовательного проекта "Урок английского" и подготовить ее для применения логики.

Задачи:

- Закрепить навык расстановки объектов на локации
- Закрепить навык тестирования своих проектов
- Изучить возможности создания переходов между сценами в рамках одного проекта
- Закрепить навыки работы с UI/UX - дизайном
- Закрепить навыки работы с логическими блоками ботов в Blockly
- Закрепить навыки настройки логики ботов

Условия победы для первой сцены

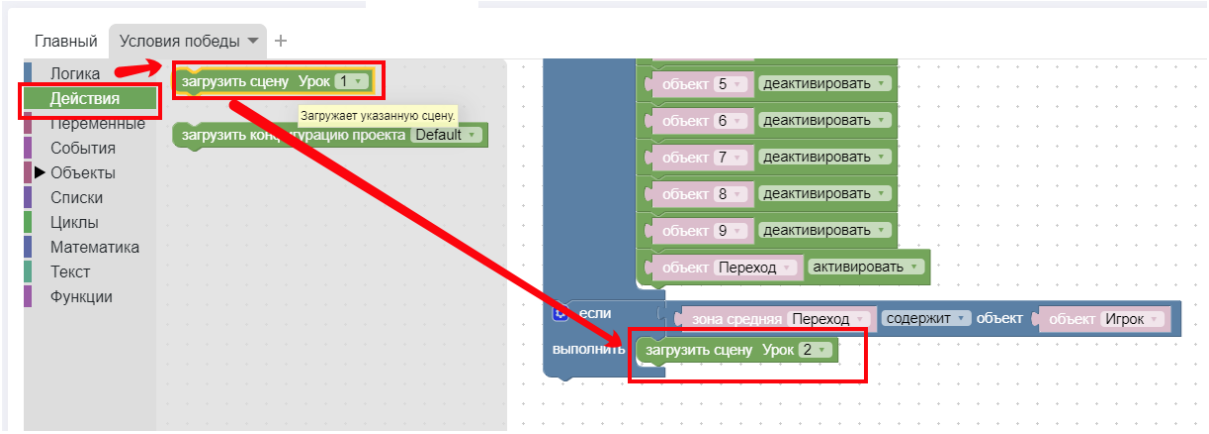
Условия победы у нас очень простые: когда результат становится больше пяти, то мы завершаем цикл случайных вопросов и Бот приглашает нас в следующую локацию.



Здесь все блоки нам уже знакомы, кроме одного - это загрузка следующей сцены.

Для начала деактивируем зону перехода, чтобы игрок не смог попасть в следующую сцену без выполнения условий, и при выполнении условий, когда результат > 5 активируем эту зону.

Необходимый блок **Загрузить сцену** находится в категории “Действия”. Просто подставляем его в тело условия при котором игрок оказывается в зоне *Перехода*.



Обратите внимание, что если в текущий момент отсутствует дополнительная сцена в структуре проекта, то вы сможете выбрать только единственный вариант, поэтому не забудьте ее создать, нажав на кнопку **Добавить сцену.**

***Совет**: если мы выполним загрузку той же сцены, на которой мы находились до этого, то это перезапустит текущую сцену и сценарий начнется заново, это может быть полезно, если по сценарию Игроку предлагается повторить задание.*

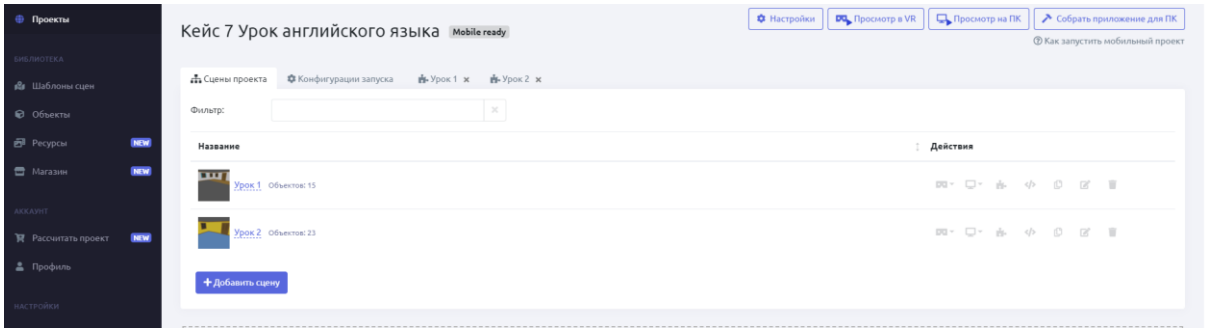
Таким образом мы полностью завершили сборку первой сцены проекта и переходим к следующей.

Сборка второй сцены

Это задание похоже на предыдущее, поэтому большую часть будет предложено выполнить самостоятельно, но здесь мы несколько усложним логику, по которой пройдемся отдельно.

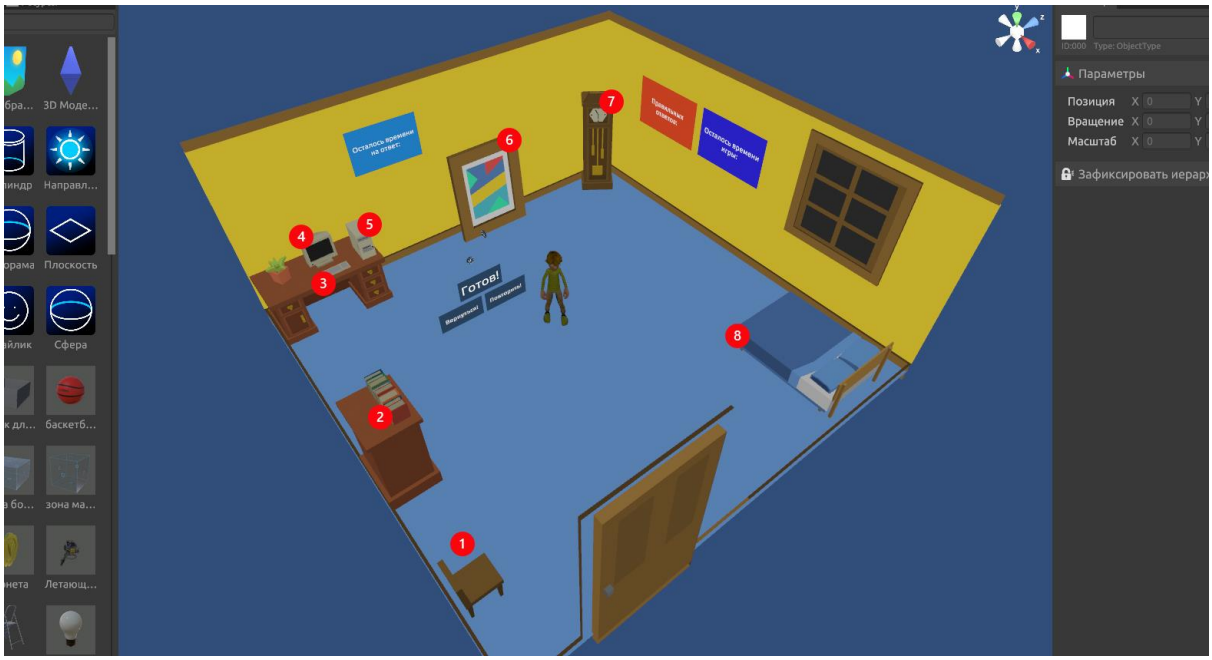
1. Во первых игроку потребуется не просто нажимать на UI с числами, а искать и использовать предметы в комнате.
2. После каждого ответа Бот не будет просто задавать следующий вопрос, но и будет реагировать на правильный или неправильный ответ.
3. Во вторых добавим элементы геймификации и введем таймер, который будет отсчитывать время на выполнение задание и таймер для ответ на вопрос Бота.

Перед началом сверимся со структурой проекта, у нас сейчас она выглядит так, для второй сцены мы выбрали локацию “Спальня”:



Сборка сцены

Разместим объекты на этой локации:



Из объектов, которые нам предстоит искать выделим интерактивные:

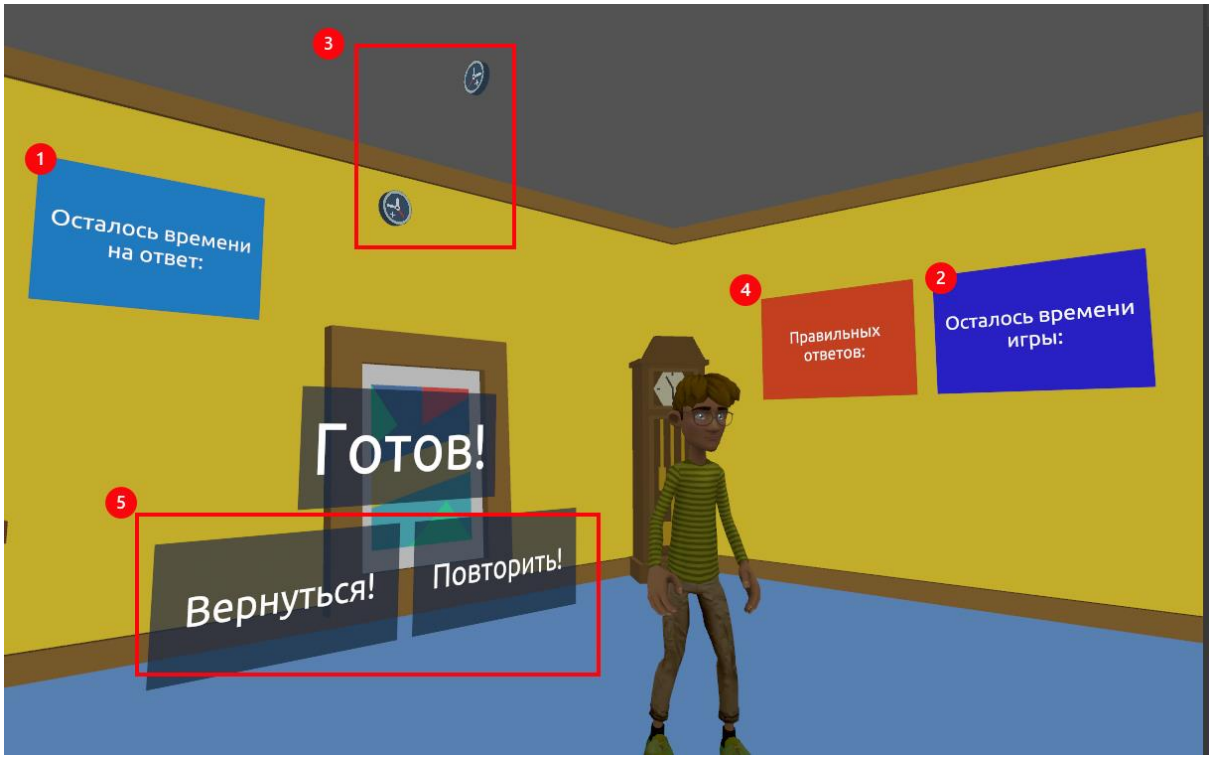
1. Стул (Chair)
2. Книги (Books)
3. Клавиатура (Keyboard)
4. Цветок (Flower)
5. Компьютер (Computer)
6. Картина (Picture)
7. Часы (Whatch)
8. Кровать (Bed)

Самостоятельная работа:

Настроим бота-мальчика, как и в предыдущем кейсе: чтобы он давал нам задание на поиск предметов с фразой: *“Теперь я буду называть предметы по-английский, а ты должен взять в руку названный предмет. И да, у тебя будет всего 30 секунд на игру и 10 секунд на ответ! Готов?”.*
Создадим UI с кнопкой “Готов!”, как и в предыдущем кейсе.

Также: *вы можете наполнить комнату дополнительными не интерактивными пропсами и другими интерактивными объектами для поиска, а также настроить поведение бота - будет ли он подбегать к нам в начале Урока или стоять на месте, при этом не забудьте задать минимальное расстояние остановки перед целевым объектом.*

Наконец, нам понадобятся (3) UI-таймеры (глобальный и таймер времени на ответ) для того, чтобы игрок знал сколько времени у него осталось на игру (1) и на ответ (2), свой результат (4) и кнопки, которые будут появляться при завершении (5) сценария:



Все необходимые объекты на локации размещены и на следующем занятии мы перейдем к настройке логики второй сцены проекта.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие логические блоки присутствуют в объекте «Бот»?
2. Как создать переход между сценами в рамках одного проекта?
3. Как перезапустить текущую сцену в процессе игры?

Занятие 6.5 Продвинутая работа с текстом и таймерами

Цель:

Реализовать логику второй сцены образовательного проекта "Урок английского", соответствующую техническому заданию.

Задачи:

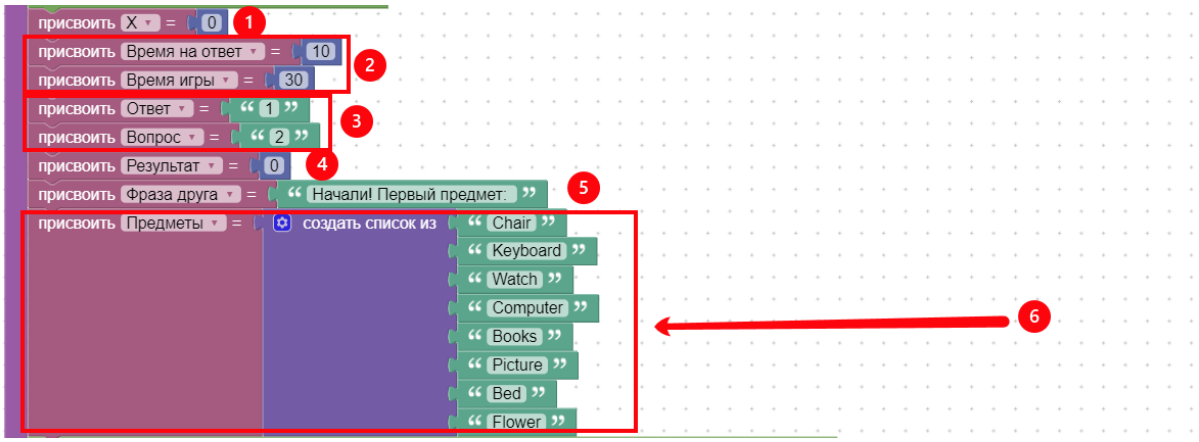
- Закрепить навыки работы с логическими блоками ботов в Blockly
- Закрепить навыки настройки логики ботов
- Закрепить навыки тестирования своих проектов
- Создать необходимые функции для реализации проекта
- Реализовать логику задания второй сцены
- Закрепить навыки работы с активацией/деактивацией объектов в Blockly
- Научиться работать с расширенным функционалом таймера
- Научиться использовать переменные внутри текста
- Научиться настраивать несколько таймеров в проекте (глобальный и локальный таймеры)

Сборка логики второго Урока

Поскольку основа логики второго урока во многом повторяет первую, то мы не будем разбирать все аспекты подробно, но остановимся подробнее на создании логики таймеров.

Объявление переменных

Нам понадобятся следующие переменные:

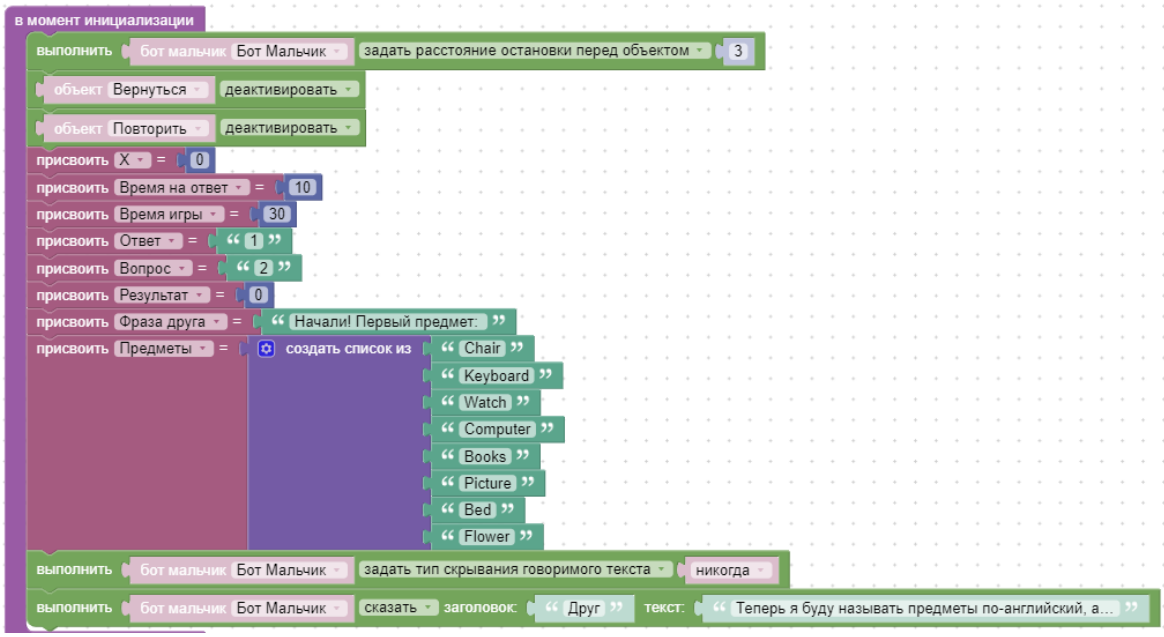


1. Переменная ***X*** - для генерации случайного числа, аналогично переменной *Случайный выбор вопроса* из предыдущей сцены.
2. Переменные ***Время на ответ*** и ***Время игры*** - в этих переменных хранятся значения для соответствующих таймеров, а именно количество их срабатываний до наступления определенного события. Для корректировки сценария мы можем быстро менять эти значения.
3. Переменные ***Ответ*** и ***Вопрос*** - аналогично предыдущему кейсу.
4. Переменная ***Результат*** - также аналогична предыдущему кейсу, будет увеличиваться на один при правильном ответе.
5. Переменная ***Фраза друга*** - в этот раз у нашего Бота будет гораздо больше текста, поэтому для оптимизации кода мы будем хранить его стандартные фразы в переменных, вы убедитесь, что это гораздо удобнее, когда мы будем собирать функции.

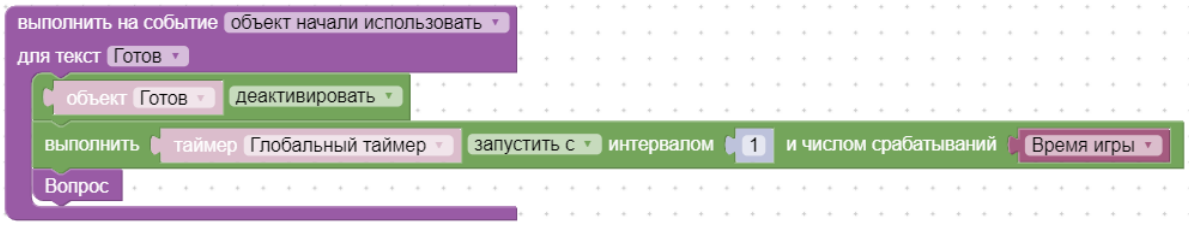
6. Наконец, список ***Предметы*** - в этом списке мы собираем названия предметов на английском языке.

Примечание: в Varwin элементы в списке индексируются автоматически, в соответствии с их расположением в Blockly по порядку сверху - вниз, начиная с единицы и далее по порядку целыми числами (1, 2, 3, 4, 5... и так далее).

В целом блок “В момент инициализации” у нас должен выглядеть так, но у вас могут быть и свои особенности:

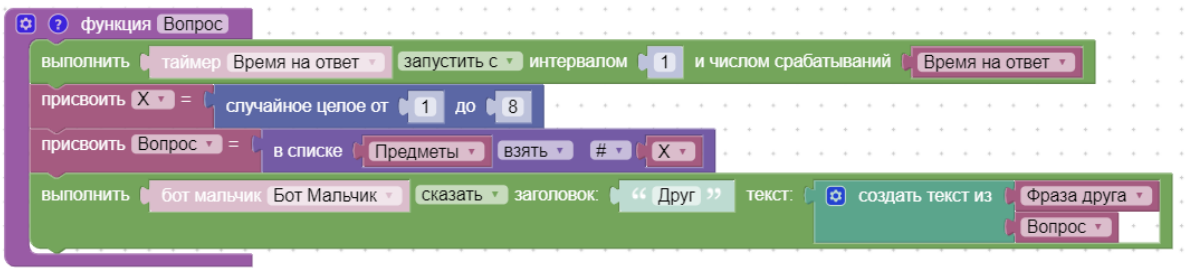


Далее сразу же собираем событие при нажатии на UI “Готов!”:



Это событие инициализирует таймер Время игры (30 секунд) и запускает функцию Вопрос.

Функция Вопрос



Итак, функция *Вопрос* активирует таймер *Время на ответ* (10 секунд) и генерирует случайное целое *X* от 1 до 8 (в соответствии с количеством индексов в списке, у вас предметов может быть больше).
Далее переменной *Вопрос* присваивается значение элемента из списка с соответствующим индексом *X*. После этого Бот произнести составную фразу из двух переменных *Фраза Друга* + *Вопрос*. Фраза друга в момент инициализации: *“Начали! Первый предмет: “* и далее предмет, случайным образом выбранный из списка *Предметы*.
Переменная Фраза Друга будет меняться в зависимости от наших ответов, без этой переменной нам бы пришлось создавать логику для каждого из новых вариантов, что в несколько раз увеличило бы сложность Blockly.

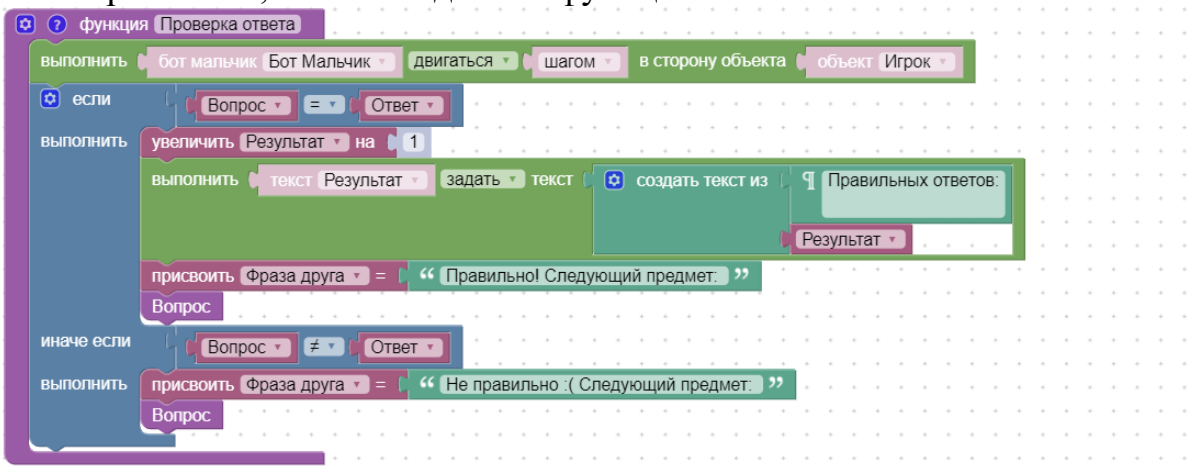
Логика выбора предмета

Далее, аналогично предыдущему кейсу создаем блоки, которые управляют выбором одного из интерактивных предметов.
Данные блоки присваивают новое значение переменной *Ответ*, в зависимости от того на какой предмет нажимает Игрок, останавливает таймер *Время на ответ*, если не прошло 10 секунд и запускает функцию *Проверка ответа.*

В отличие от предыдущего Урока мы не запускаем сразу функцию Следующий вопрос, поскольку иначе было бы сложно реализовать реакцию Бота на наш ответ, поэтому реализуем это в теле функции Проверка ответа.

Функция проверка ответа

Посмотрим на то, как выглядит эта функция:

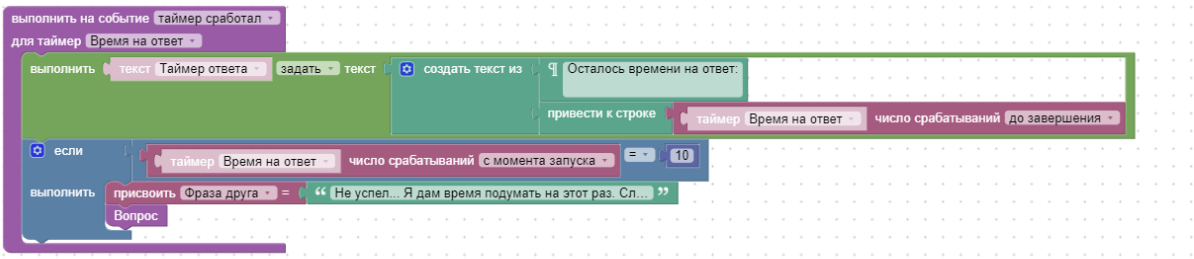


Во первых, каждый раз, когда мы нажимает на предмет Бот будет к нам побегать, чтобы задать следующий вопрос, это удобно с точки зрения UX-дизайна, чтобы всякий раз не подбегать к Боту для получения следующего задания.

Далее все очень просто:

1. Если *Вопрос* = *Ответ*, то мы увеличиваем *Результат* на единицу, обновляем значение на UI Результат, присваиваем *Фразе Друга* значение: *“Правильно! Следующий предмет: “* и активируем функцию следующий *Вопрос*.
2. Иначе, если *Вопрос* **не равен** *Ответу*, то *Результат* не увеличивается и *Фраза друга* становится иной.

Наконец, возможен еще один дополнительный вариант, когда мы не успеваем найти предмет за 10 секунд, для этого случая соберем отдельное событие:



Мы запускаем таймер *Время на ответ* каждый раз, когда активируется функция *Вопрос* с интервалом в 1 секунду. Т.е.таймер срабатывает каждую секунду и мы должны проверять определенные условия при каждом срабатывании.

Первый блок отвечает за обновление UI Таймер ответа при каждом срабатывании.

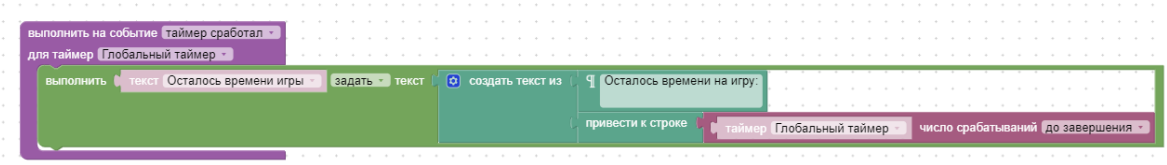
Блок с условным оператором проверяет, что число срабатываний с момента запуска меньше 10, если же это условие наступает, что Фраза друга становится иной: “Не успел... Я дам время подумать на этот раз. Следующий предмет: “ и запускается функция *Вопрос*.

Как видите, мы предусмотрели все варианты развития событий.

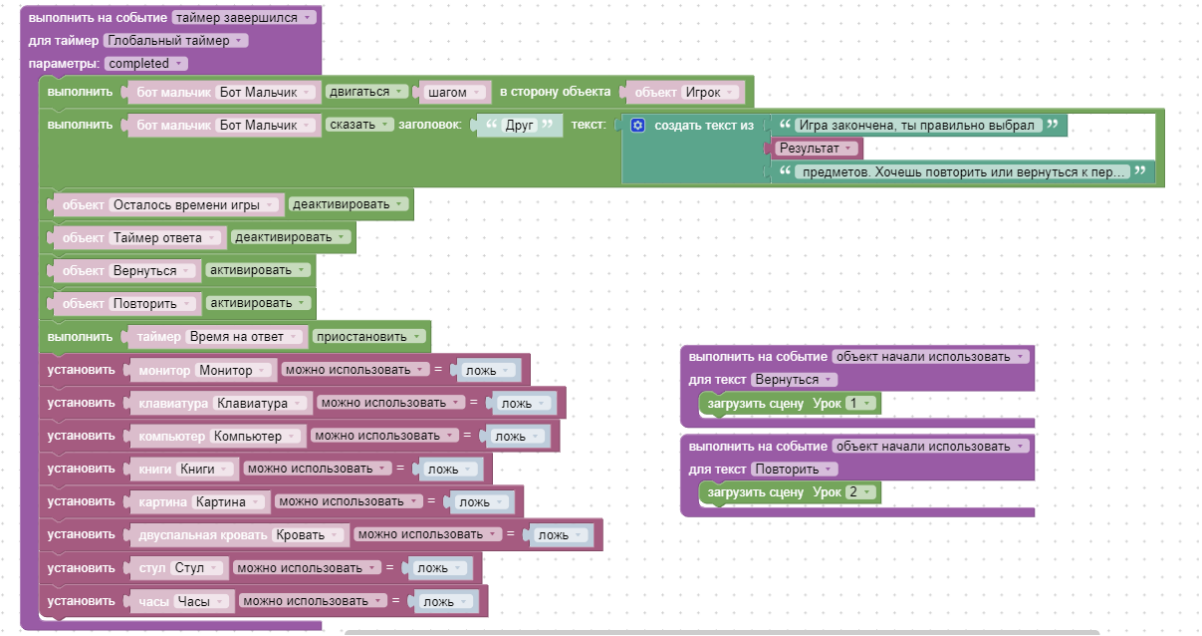
***Примечание**: если бы не хранили текст Фраза Друга в переменной, то нам пришлось бы как минимум создавать три функции Вопрос с различными вариантами фраз, вместо одной (Вопрос, если предыдущий ответ правильный, Вопрос, если предыдущий ответ неправильный и Вопрос, если мы не успели), согласитесь, что это было бы менее практично? Постарайтесь найти применение этой практике в своих будущих проектах!*

Глобальный таймер и финал игры

Соберем событие, которое будет отвечать за обновление UI Осталось времени игры:

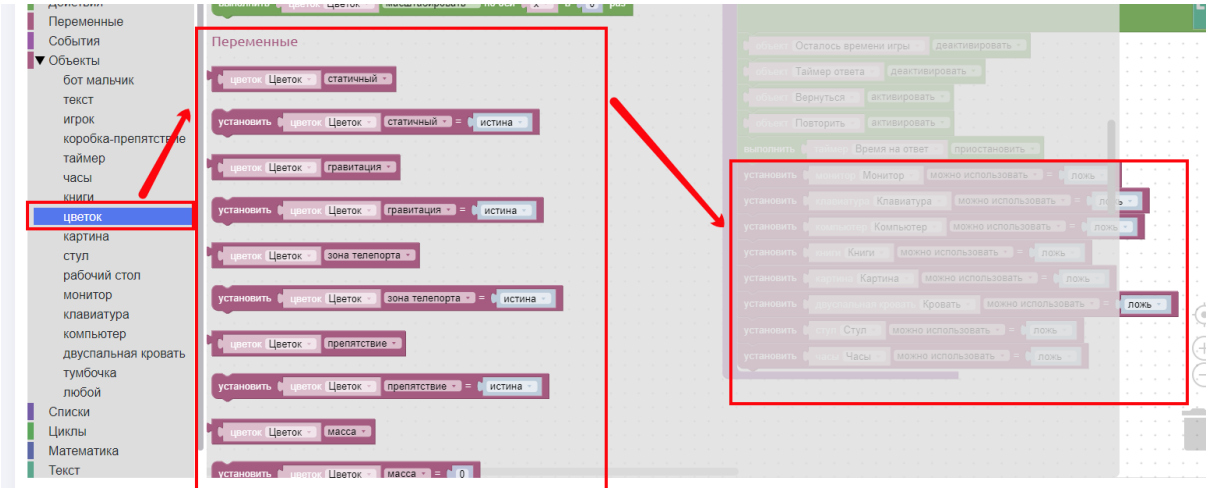


В этот раз игра заканчивается, когда заканчивается глобальный таймер, поэтому для финала игры у нас должно быть создано следующее событие:

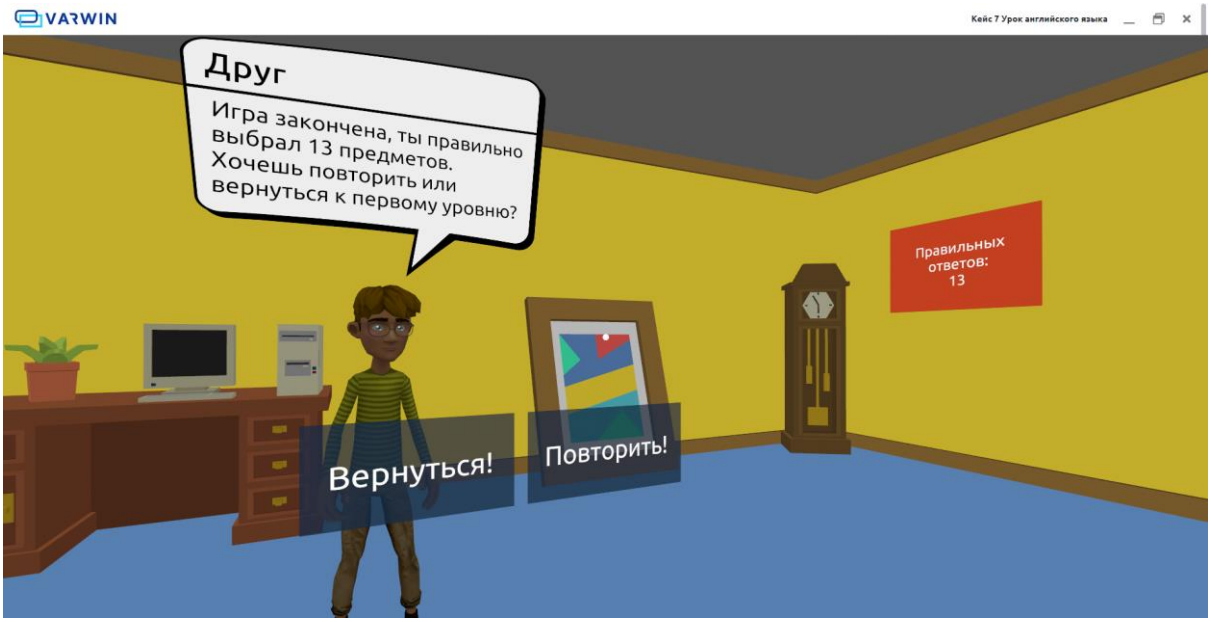


***Самостоятельная работа:** все, что есть в этом финальном блоке мы уже ранее разбирали, поэтому для вас не должно составить особого труда собрать условия для финала игры. Можно заметить, что после завершения глобального таймера мы запрещаем игроку возможность использовать предметы, чтобы он не сломал логику приложения, а также даем возможность либо повторить текущий сценарий, либо вернуться к предыдущему Уроку с числительными - таким образом, мы зацикливаем все наше приложение, что может быть удобно с точки зрения UX/UI-дизайна.*

***Примечание**: как мы уже разбирали до этого, свойства объектов, которые мы можем настраивать в инспекторе можно также менять через Blockly в разделе Переменные:*



Если вы все сделали правильно, то в финале вы увидите примерно такой экран. Поздравляем с завершением работы над еще одним кейсом, впереди нас ждет финальный самый сложный и интересный проект!



Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какой блок из категории «Текст» обеспечивает перенос на следующую строку?
2. Где находится блок «Активации/Деактивации объекта»?
3. Какие логические блоки есть у объекта таймер в Blockly?

Занятие 6.6 Расширение проекта урок Английского Языка

Цель:

Усвоение навыков, полученных в ходе практических занятий. Развитие (апгрейд) проекта “Урок английского” по собственному техническому заданию.

Задачи:

- Закрепить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Повысить навыки рисования скетчей/ планов перемещения по виртуальному пространству
- Закрепить навыки создания и использования дизайна интерфейсов, основанном на удобстве для конечного пользователя
- Усвоить навыки работы с функциями в Varwin
- Усвоить навыки работы с таймером
- Усвоить навыки работы с расширенным функционалом текста

Формат: Самостоятельная работа, решение кейсового задания.
Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

0 - 10 минут	Получение кейса. Формирование собственного технического задания по проекту "Урок английского".

На этом этапе обязательно нужно сформировать и зафиксировать техническое задание на проект

Шаблон технического задания:

1. Тематика/направленность проекта;
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы или же свои объекты);
3. Основной функционал приложения;
4. Дополнительные возможности приложения;
5. План-схема сцен, которые будут реализованы;
6. Каким образом будет реализован, удобный для пользователя, UX/UI - дизайн;
7. Какую полезную функцию для общества несет проект, или будет нести при дальнейшей его доработке.

Рекомендация: если есть возможность, то можно посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом между обучающимися.

Кейс:

Необходимо расширить проект “Урок английского”. Сейчас вам предоставляется творческая свобода и Вам решать как расширять этот проект. Вы можете создать дополнительную мини-игру, например по расстановке букв в слова или угадывать пропущенные буквы в словах. Можете расширить текущие сцены, например во второй сцене отсортировать угаданные объекты по разным категориям. Самое главное соблюдайте обязательные условия.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта
2. Нарисовать план расположения объектов на сцене
3. Зафиксировать дополнительные функции, которые будут реализованы в проекте
4. Реализовать минимум два списка в рамках одной сцены
5. Использовать 3D-объекты для новых функций

7 Циклы

Занятие 7.1 Циклы

Цель:

Познакомиться с понятием “цикл”, узнать что это такое, для чего предназначено и как использовать в своих проектах на Varwin.

Задачи:

- Сформировать понимание определения “цикл” в программировании
- Узнать какие основные типы циклов существуют и как они работают
- Познакомиться с логическими блоками циклов в Blockly
- Рассмотреть ситуации в которых можно использовать циклы
- Изучить возможности остановки циклов
- Обсудить возможности применения циклов в проектах на Varwin

Методические материалы для подготовки к занятию:

Литература для подготовки к занятию:

Цикл (программирование) — Википедия

Урок 6. Циклы. Какие бывают циклы. (Что такое for,while,do while)

Определение:

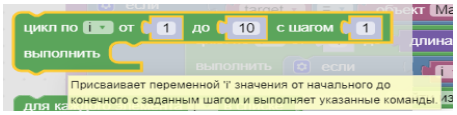
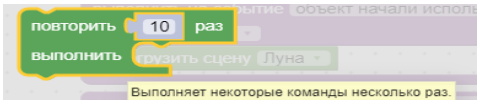
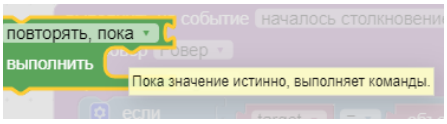
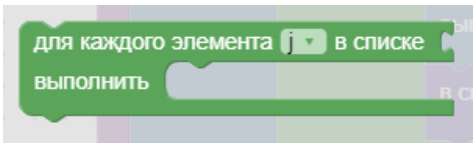
Цикл - это последовательность операторов, которая выполняется несколько раз, пока выполняется (или ие выполняется) определенное условие или заранее заданное количество раз. Выполняемая в цикле последовательность операторов называется **телом цикла**. Каждое прохождение цикла называется **итерацией.**

Циклы бывают трех основных видов:

For	While	do While
Цикл со счетчиком. Необходимо использовать когда мы знаем количество итераций цикла.	Цикл с предусловием. Будет выполняться пока условие верно.	Цикл с постусловием. Будет выполняться первый раз и проверит условие и если оно верно выполняется снова, и так пока условие будет верно

В Varwin Циклы в основном используются для перебирания элементов из Списков, с определенной целью.

В XRMS Varwin существуют логические блоки для работы с циклами, которые мы сейчас с Вами сформируем в таблице. Эту таблицу Вы в дальнейшем сможете использовать для справки:

Название цикла	Описание	Изображение
Цикл по **i**	Перебирает все элементы цикла до наступления определенного условия для i	
Повторить **X** раз	Повторяет действия в теле цикла определенное количество раз. Это количество может быть задано целым числом или переменной целого типа.	
Повторять Пока **<Условие >**	Повторяет действия в теле цикла, пока не наступит выполнение определенного условия. Например, это может быть значение переменной, которая становится больше какого-то значения.	
Для каждого элемента **j** в **<Список>**	Перебирает последовательно все элементы списка, пока эти элементы не закончатся.	
Выйти из цикла	При выполнении определенного условия заканчивает выполнение действий в теле цикла.	

На этом этапе нужно поработать с этими логическими блоками и внимательно посмотреть как они работают. Важно усвоить понимание работы разных циклов, для дальнейшего применения их в своих проектах в нужные моменты.

Особое внимание стоит уделить блокам “выйти из цикла” и “перейти к следующему шагу цикла”. Они позволяют в нужный момент прервать выполняющийся цикл любого типа или перейти к следующему шагу цикла.

Самостоятельная работа: подумайте где, как и для чего мы можем применять циклы? Вспомните как мы использовали механики циклов в прошлых проектах.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Что такое цикл?
2. Какие основные типы циклов существуют?
3. В какой момент можно остановить выполнение цикла и как это сделать?
4. Чем блок «выйти из цикла» отличается от блока «перейти к следующему шагу»?

Занятие 7.2 Сборка сцены Луна

Цель:

Разработать сцену образовательного проекта "Космическая миссия" и подготовить ее для применения логики, инициализировать переменные.

Задачи:

- Закрепить навык расстановки объектов на локации
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с UI/UX - дизайном
- Закрепить навыки работы с логическими блоками ботов в Blockly
- Закрепить навыки работы с вкладками логики в Blockly
- Закрепить навыки работы с техническим заданием
- Закрепить навыки работы с переменными и их инициализацией

Техническое задание:

Необходимо создать приложение для Varwin, которое будет воспроизводить лунную миссию, состоящую из трех шагов:

1. Подвести ровер к маяку для получения координат (избегайте столкновений!).
2. Вычислить оптимальный угол радара для отправки сообщения.
3. Установить контакт с инопланетной цивилизацией, повернув радар на оптимальный угол.

Специальные требования:

1. Препятствия должны генерироваться случайным образом при каждом запуске приложения.
2. Должен быть реализован интерфейс для управления ровером.
3. Требуется реализовать шкалу здоровья для ровера, которая будет уменьшаться при столкновении с препятствиями. При обнулении шкалы здоровья миссия должна начинаться сначала.
4. Оптимальный угол должен рассчитываться автоматически из случайно сгенерированных координат и быть равен среднему арифметическому координат.
5. UI должен информировать игрока об основных шагах выполнения миссии и о сгенерированных и вычисленных данных.

6. Радар должен поворачиваться на вычисленный оптимальный угол, который задается случайным образом при каждом запуске приложения.

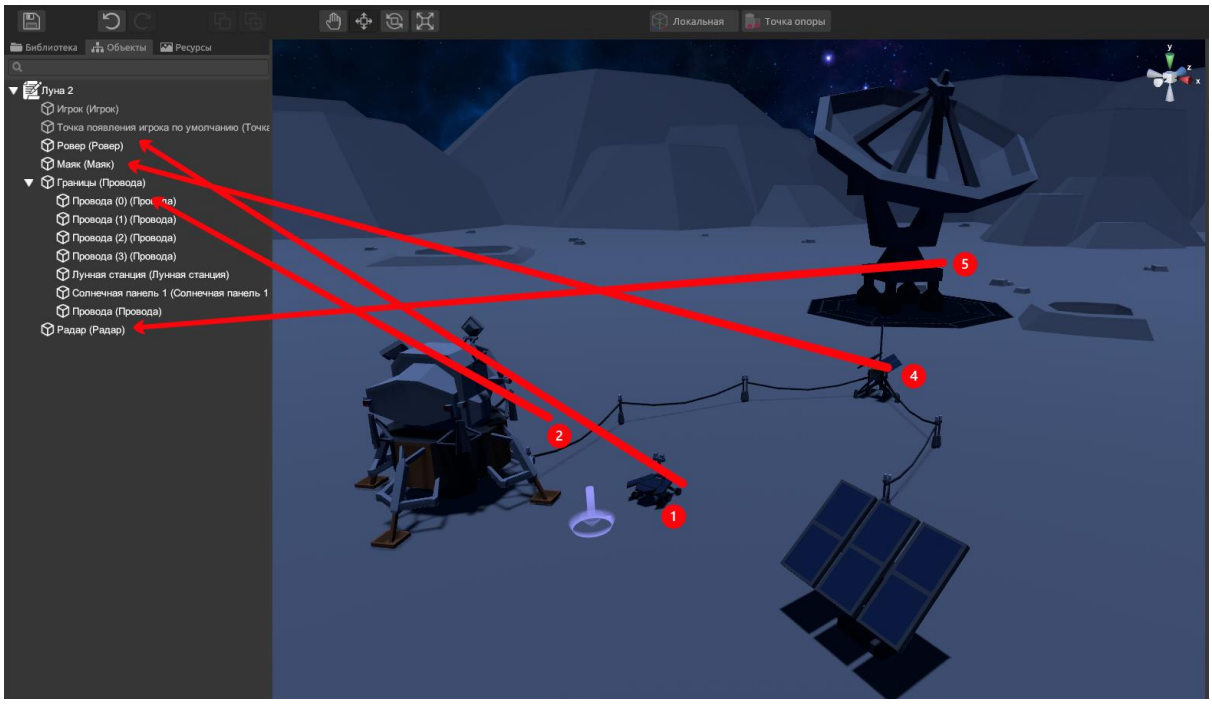
Подробно и поэтапно разберите с обучающимися техническое задание. Это финальный кейс, он довольно сложный и чтобы его реализовать нужно максимально внимательно подойти к занятиям. Проект содержит много разной логики и вкладок с логикой. **Обязательно** структурируйте свою логику в разные функциональные вкладки.

Сборка сцены Космической миссии

В качестве основной сцены в нашем случае выбрана локация “Луна.
Разместим на ней основные объекты:

1. **Ровер**, который будет двигаться к маяку.
2. С помощью **Проводов** зададим границы перемещения ровером в целях создания добного UX-дизайна. Для удобства создадим для проводов *Иерархию*.
3. **Радар**, который будет поворачиваться на *Оптимальный угол* для установления контакта с инопланетной цивилизацией.
4. И **Маяк**, к которому необходимо будет подвести ровер для получения координат.

Самостоятельная работа: *разместите на локации дополнительные пропсы, которые будут служить ограничителями игровой зоны, при этом они должны выглядеть естественно. В нашем случае, это лунная станция и солнечная батарея. Но вы можете найти больше объектов из этой тематики в пакете “Астрономия”.*

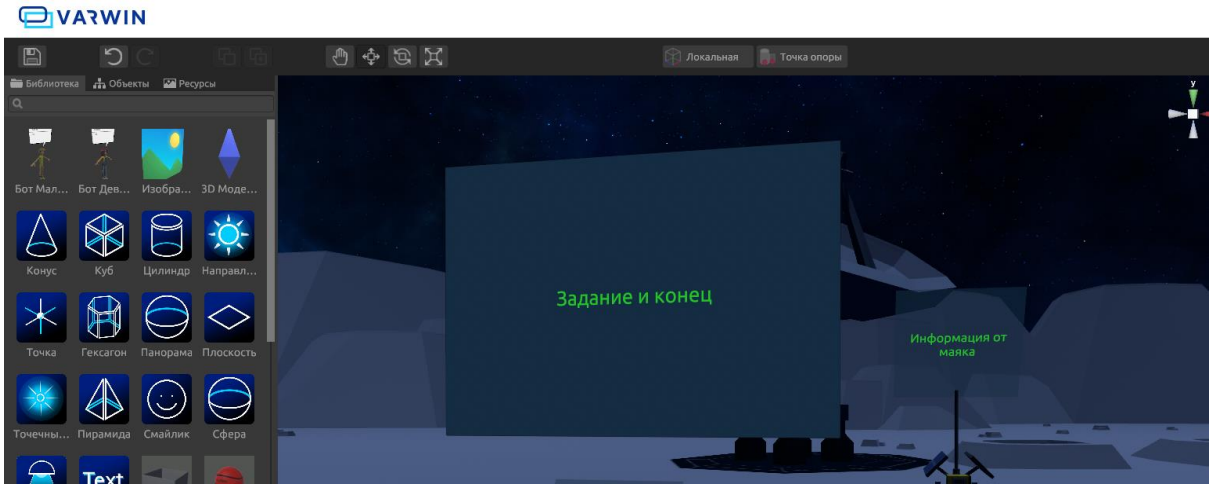


***Совет**: плохой пример UX-дизайна для ограничения игрового пространства - это использование невидимых стен, старайтесь избегать такого метода ограничения перемещения игрока, лучше использовать в качестве препятствий объекты, вписывающиеся в тематику локации.*

Далее нам необходимо разместить на сцене препятствия, которые должен будет объехать Ровер, управляемый игроком, чтобы добраться до маяка для генерации Координат.

В качестве препятствий мы использовали бочки и контейнеры, вы можете разместить препятствия по своему вкусу и сформировать для них Иерархию.

Наконец, разместим на сцене UI, которые будут отвечать за постановку задания и окончание миссии (UI “*Конец*”) и информацию от маяка (UI “*Координаты*”):

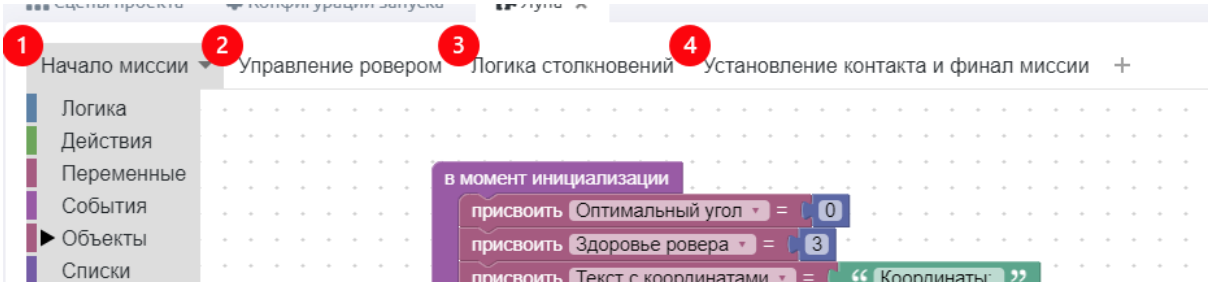


На самом деле нам еще понадобятся дополнительные объекты на этой сцене, которые будут отвечать за интерфейс взаимодействия с ровером и маяком, но мы дополним этими UI сцену в момент, когда дойдем до сборки логики этих

этапов выполнения задания, чтобы сформировать целостную картину сборки данного кейса.

Сборка логики Космической миссии

Создадим вкладки в редакторе логики для реализации каждой из основных механик, согласно ТЗ:



Из ТЗ: *Препятствия должны генерироваться случайным образом при каждом запуске приложения.*

Во первых создадим вкладку **Начало миссии (1)**, в которой опишем то, что происходит в *Момент инициализации* и условия старта. Одним из интересных условий является случайная генерация препятствий при начале миссии в *Момент инициализации*, здесь мы впервые столкнемся с циклами, поэтому сделаем на этом отдельный акцент.

Из ТЗ: *Подвести ровер к маяку для получения координат (избегайте столкновений!).*
Должен быть реализован интерфейс для управления ровером.

Для этих пунктов создадим вкладку **Управление ровером (2)**, где соберем основные события и функции для интерфейса управления.

Из ТЗ: *Подвести ровер к маяку для получения координат (избегайте столкновений!).*
Требуется реализовать шкалу здоровья для ровера, которая будет уменьшаться при столкновении с препятствиями. При обнулении шкалы здоровья миссия должна начинаться сначала.

Все, что описано в данных пунктах относится к столкновениям ровера с препятствиями или целевым объектом (*Маяк*), выделим для этой логики отдельную вкладку **Логика столкновений (3).**

Из ТЗ: *Вычислить оптимальный угол радара для отправки сообщения.*
Установить контакт с инопланетной цивилизацией, повернув радар на оптимальный угол.

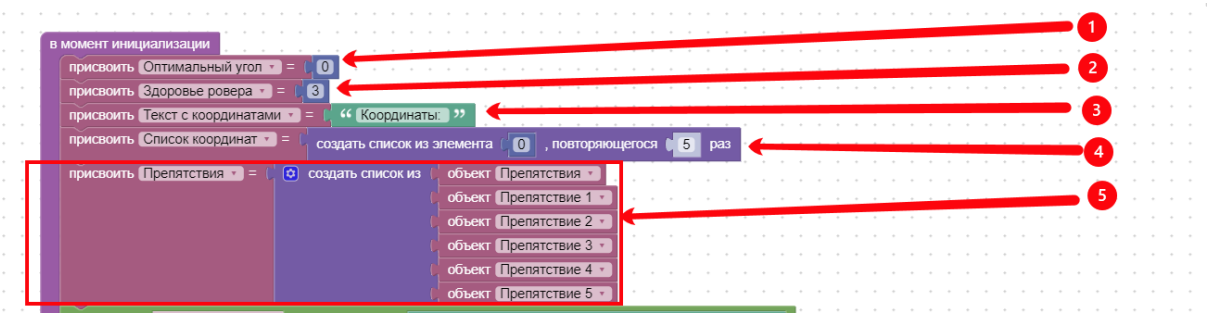
Оптимальный угол должен рассчитываться автоматически из случайно сгенерированных координат и быть равен среднему арифметическому координат.
UI должен информировать игрока об основных шагах выполнения миссии и о сгенерированных и вычисленных данных.
Радар должен поворачиваться на вычисленный оптимальный угол, который задается случайным образом при каждом запуске приложения.

Все, что описано в данных пунктах так или иначе сводится к финалу миссии - это вычисление *Оптимального угла*, интерфейс управления *Маяком* и *Радаром*, для вычисления *Оптимального угла* и поворота *Радара*, а также UI для информирования игрока.

Итак, начнем собирать логику в данном порядке - по вкладкам.

Вкладка Начало миссии
Для начала объявим переменные, это важно сделать в *Момент инициализации*, поэтому здесь мы немного зайдем на территорию механик, которые будут описаны в последующих вкладках.

***Примечание**: в процессе разработки сложного приложения часто приходится возвращаться к началу, чтобы объявить новые переменные, которые требуются для реализации логики и часто сложно превентивно понять, какие именно переменные нам понадобятся. Но поскольку мы разбираем уже собранное приложение, то для наглядности мы упростим этот момент. Однако в будущем не забывайте, что часто сложно спрогнозировать конечную архитектуру вашего приложения.*

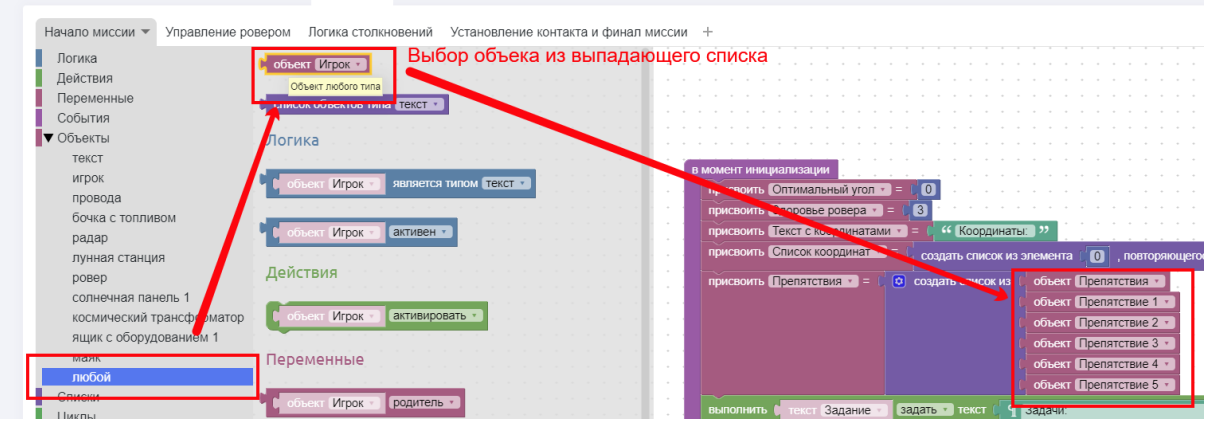


1. Переменная ***Оптимальный угол*** - это переменная будет вычисляться в процессе выполнения логики приложения, пока зададим ей значение по умолчанию “0”.
2. Переменная ***Здоровье ровера*** - эта переменная отвечает за максимальное количество столкновений *Ровера* с препятствиями, до его разрушения и перезапуска приложения.
3. Переменная ***Текст с координатами*** - это переменная, которая будет отвечать за отображение случайно сгенерированных координат на UI “Координаты”.

4. ***Список Координат*** - с помощью этой переменной мы создаем список с одинаковыми элементами, чтобы в будущем его перезаполнить случайно сгенерированными числами. Очень важно на старте определиться с длиной списка, независимо от элементов, которые будут его наполнять, зато в будущем мы сможем быстро поменять количество этих элементов одним кликом для оптимизации нашего приложения.
5. ***Список Препятствий*** - здесь мы формируем список *Препятствий*, которые ранее расставили на локации (в нашем случае это бочки и контейнеры, но у Вас может быть другой список этих объектов).

На пункте 5 остановимся подробнее.

Как вы видите, мы можем формировать списки не только из абстрактных чисел или текста, но и из объектов, которые находятся на сцене.
Для того, чтобы это сделать выберите категорию объектов “Любой”, а далее назначьте необходимый объект из выпадающего списка:



Этот список нам понадобится для создания случайной генерации препятствий в начале Космической миссии. С этого и начнём следующее занятие.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Из каких объектов можно создавать списки в Varwin?
2. Что делает блок «в момент инициализации»?
3. Что нужно сделать чтобы сформировать список из объектов?

Занятие 7.3 Создание случайных препятствий

Цель:

Разработать для проекта "Космическая миссия" вкладку логики “Начало миссии”.

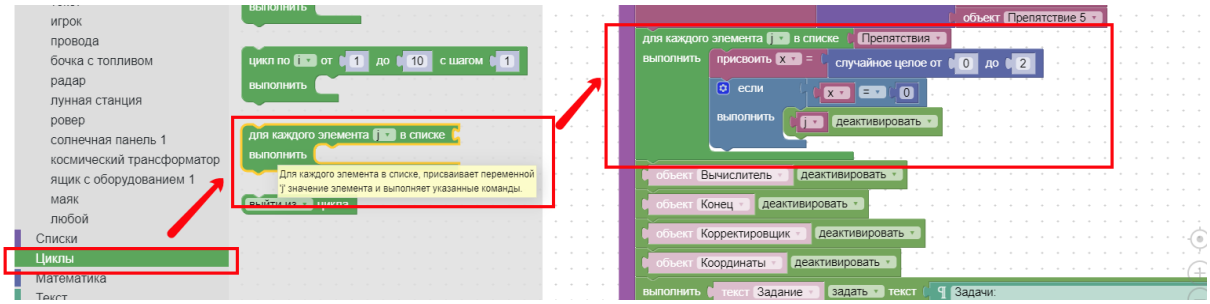
Задачи:

- Научиться перебирать объекты в списках
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с логическими блоками в Blockly
- Закрепить навыки работы с активацией/деактивацией объектов
- Закрепить навыки работы с продвинутыми функциями текста

Что такое циклы и генерация случайных препятствий

В нашем случае необходимо перебрать элементы из списка Препятствия, чтобы определить какие из препятствий будут активированы, а какие деактивированы на сцене случайным образом.

Для этого мы используем логический блок из категории “Циклы” Для каждого элемента списка X.



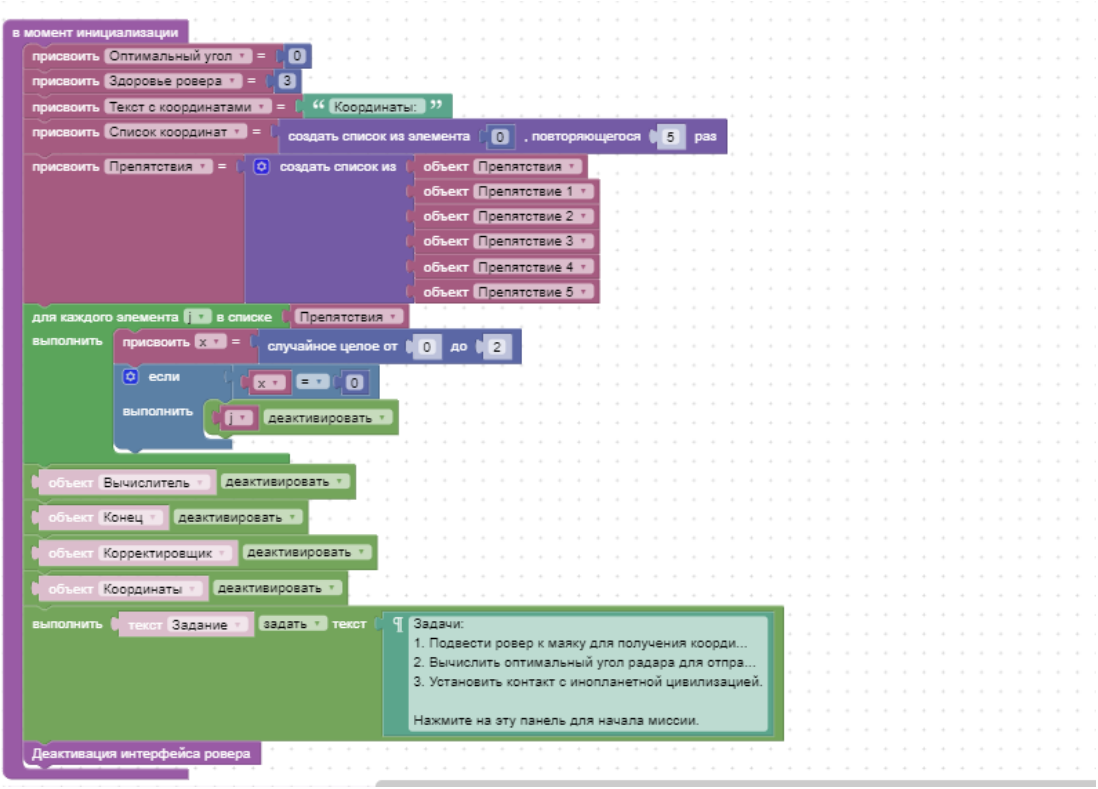
Этот блок последовательно перебирает все элементы из списка Препятствия и заканчивает работать, когда элементы из списка заканчивается.

Для каждого элемента генерируется случайное число от 0 до 2, при этом, если это сгенерированное число = 0, то мы деактивируем этот объект.

Таким образом при каждом запуске приложения мы получаем разный состав из препятствий для Ровера.

Завершения создания вкладки Начало миссии

Финальный результат должен выглядеть так:



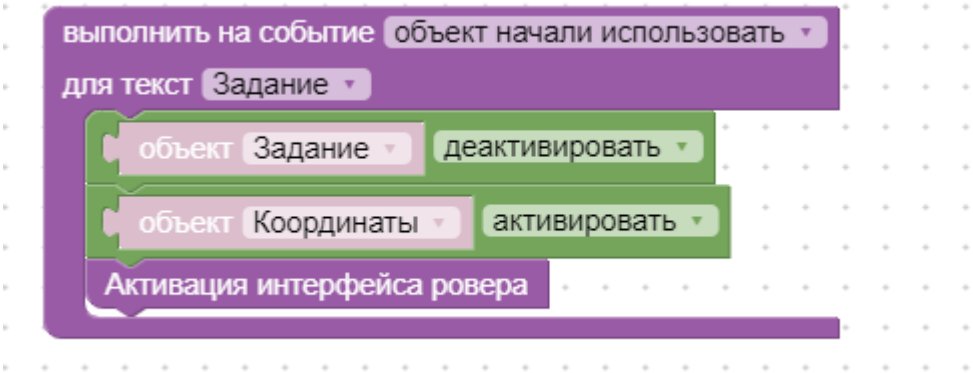
Здесь используются элементы интерфейса, которые мы еще не разместили на локации, поэтому, если вы видите незнакомые названия, то вернитесь к этому скриншоту позже (это относится к блокам из интерфейсов управления *Маяком* и *Радаром (“Вычислитель”, “Корректировщик”*).

Главное, что необходимо сделать в текущий момент - это установить текст для UI “*Задание*”:

Задачи:
1. Подвести ровер к маяку для получения координат (избегайте столкновений!).
2. Вычислить оптимальный угол радара для отправки сообщения.
3. Установить контакт с инопланетной цивилизацией.
Нажмите на эту панель для начала миссии.

Напомним, что UI “Координаты” отвечает за информацию от *Маяка*.

Кроме этого, мы видим новую функцию **Деактивация интерфейсов ровера**.



Далее нам необходим Блок, который будет управлять логикой старта задания, в момент использования Игроком UI “Задание” активируются функция **Активации интерфейсов ровера**, эту функцию наряду с **Деактивацией интерфейсов ровера** мы разберем в следующей вкладке.

Занятие 7.4 Создание интерфейса управления луноходом

Цель:

Разработать для проекта "Космическая миссия" вкладку логики “Управление ровером”.

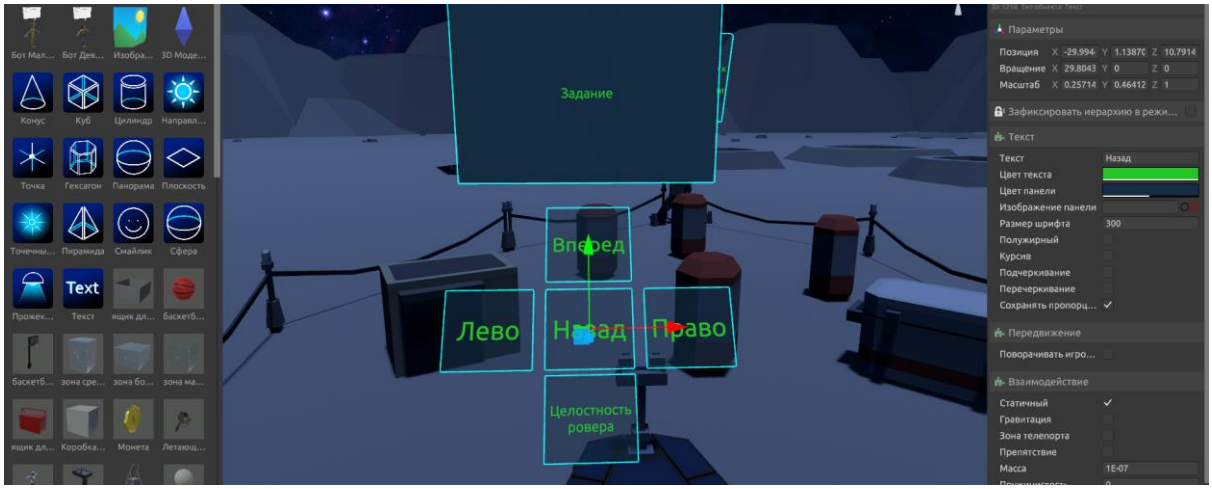
Задачи:

- Реализовать интерфейс и логику управления ровером
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с логическими блоками в Blockly
- Закрепить навыки работы с активацией/деактивацией объектов
- Закрепить навыки работы с продвинутыми функциями текста
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с UI/UX - дизайном
- Закрепить навыки работы с событиями “Объект начали использовать”

Вкладка Управление ровером

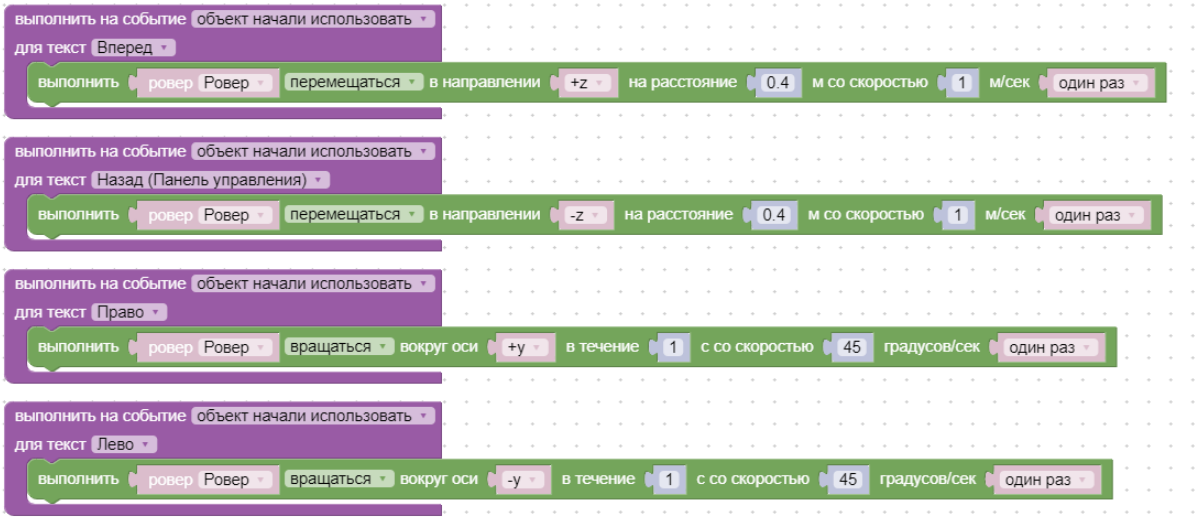
В этой вкладке мы настроим управление ровером и интерфейсы управления.
Для этого для начала разместим на сцене интерфейсы ровера из объектов “Текст” (*Вперед, Назад, Лево, Право, Целостность ровера*):

***Примечание**: UI “Целостность ровера” мы разберем подробнее на следующей вкладке, пока просто добавьте этот интерфейс.*

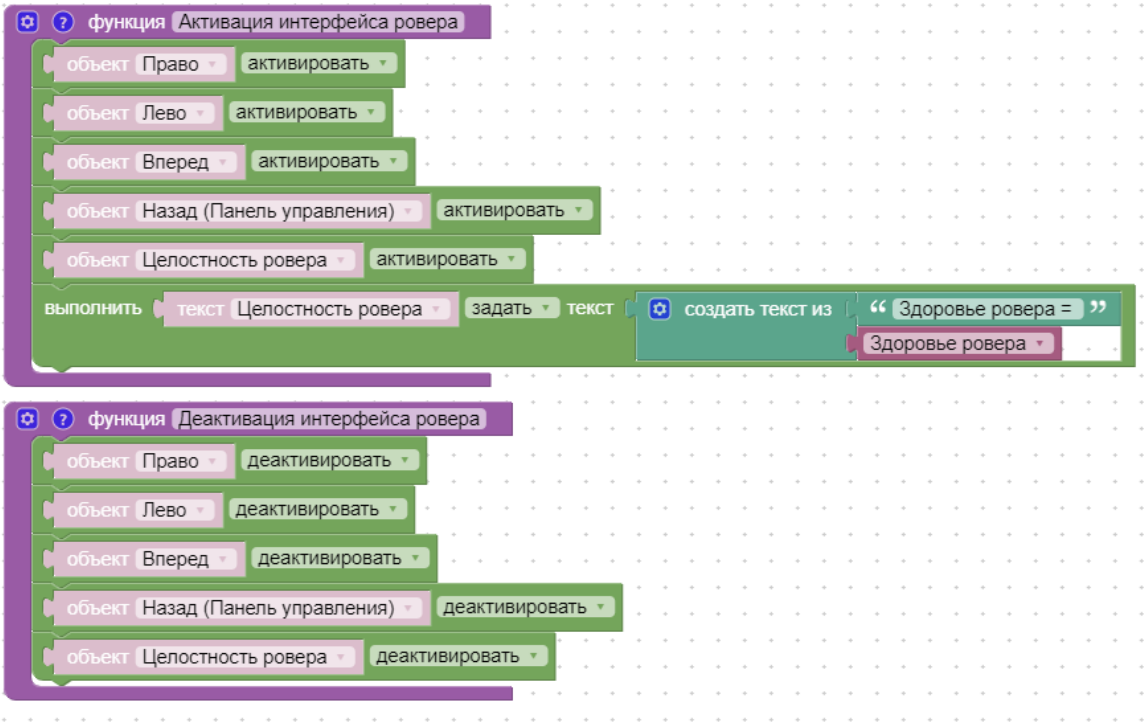


Далее на вкладке Управление ровером создадим логику использования этих UI.

Самостоятельная работа: *логику построения последующих событий мы уже разбирали ранее в кейсе №5 "Реконструкция битвы", воспользуйтесь таблицей стандартной логики, чтобы вспомнить блоки стандартной логики объектов и реализуйте интерфейс управления Ровером как на картинке ниже. Чтобы определить наилучшую скорость и угол поворота тестируйте приложение.*



Дополнительно нам необходимо реализовать функции активации и деактивации интерфейса управлением ровера, чтобы использовать их в необходимый момент сценария:



Это необходимо для удобной реализации UX/UI-дизайна, когда интерфейсы видимы для Игрока только в нужный момент сценария. В начале миссии (до активации UI “Здания” эти интерфейсы нам не нужны, поэтому мы используем функцию **Деактивации интерфейсов ровера** и используем **Активацию интерфейсов ровера**, когда игрок нажал на UI “Задание” для начала задания.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Как правильно называть объекты так, чтобы в событиях они отображались не полностью, а только необходимая для опознавания часть названия?
2. Какой блок необходимо использовать чтобы задать расстояние и скорость перемещения объекта?
3. Как работает блок “создать текст из”?

Занятие 7.5 Математика

Цель:

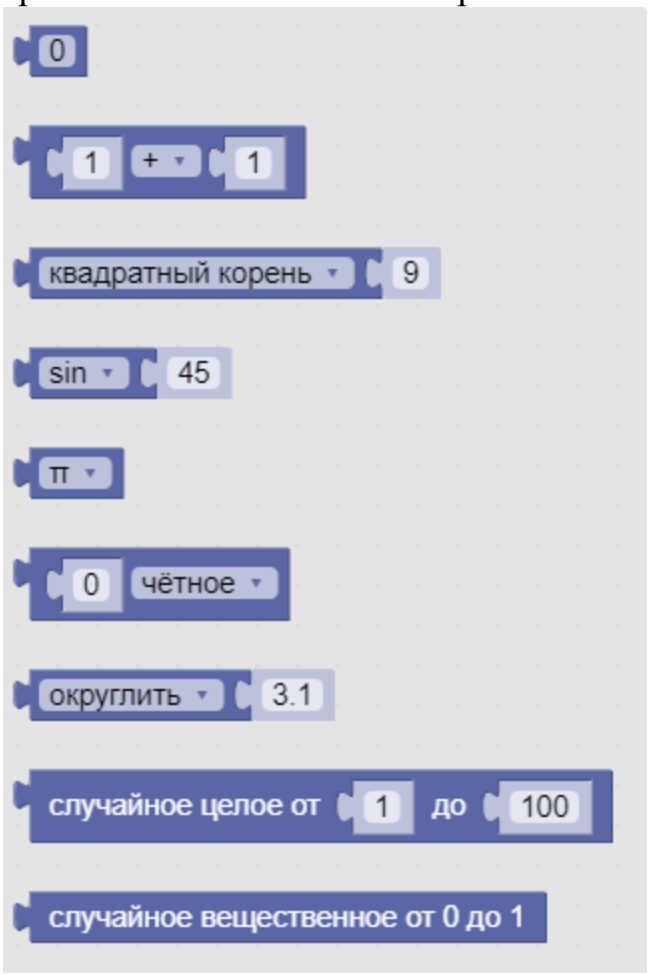
Познакомиться с логическими блоками категории математика в XRMS Varwin и понять для чего их можно использовать.

Задачи:

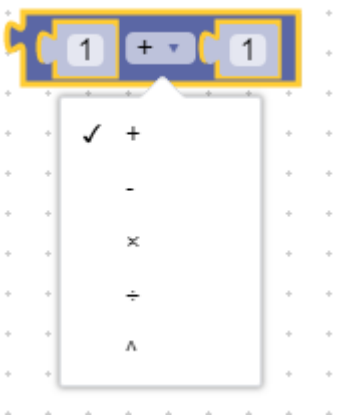
- Сформировать понимание математики в Varwin
- Поработать с логическими блоками категории математика
- Рассмотреть ситуации в которых можно использовать математические блоки

Методические материалы для подготовки к занятию:

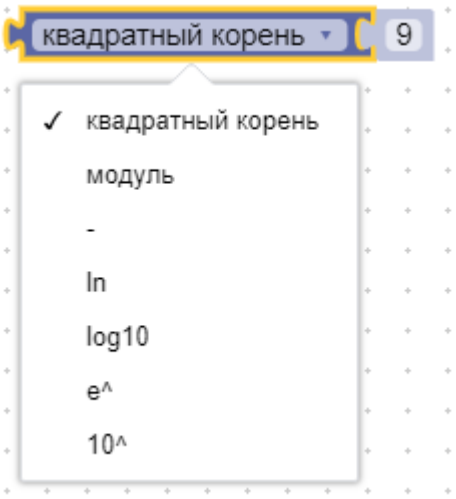
А сейчас мы рассмотрим основные блоки категории математика в Varwin:



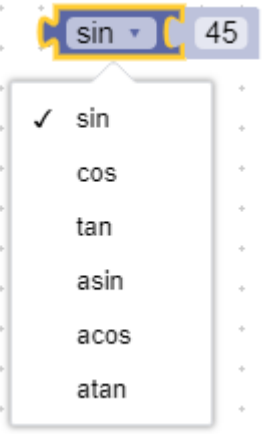
Так выглядит вся категория непосредственно в Blockly. Здесь есть выбор простого числа, выполнение простейших математических операций между двумя числами, выборка случайного числа от и до. Но давайте посмотрим их поближе, потому что в математике очень много функций в выпадающих списках.



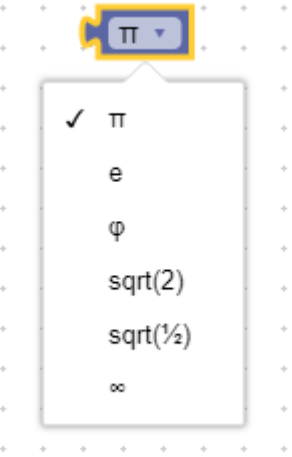
Например, у блока простейших операций есть сложение, вычитание, умножение, деление и возведение в степень.



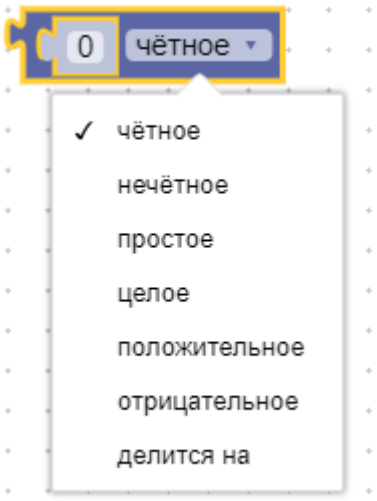
У функционального блока присутствуют квадратный корень, модуль, отрицание, натуральный логарифм, десятичный логарифм, возведение “e” в степень, возведение “10” в степень.



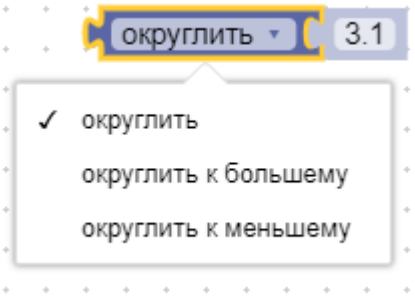
Тригонометрические функции: синус, косинус, тангенс, арксинус, арккосинус, арктангенс.



Постоянные числа: Пи, e, фи, квадратный корень из 2, квадратный корень из ½, бесконечность.



Типы чисел: четное, нечетное, простое, целое, положительное, отрицательное, делится ли на.



Округление чисел: просто округление, округление к большему или меньшему.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Какие есть логические блоки в категории «Математика»?
2. Что будет с числом 168.3434 если его округлить к меньшему?

Занятие 7.6 Настройка столкновений

Цель:

Разработать для проекта "Космическая миссия" вкладку логики “Логика столкновений”.

Задачи:

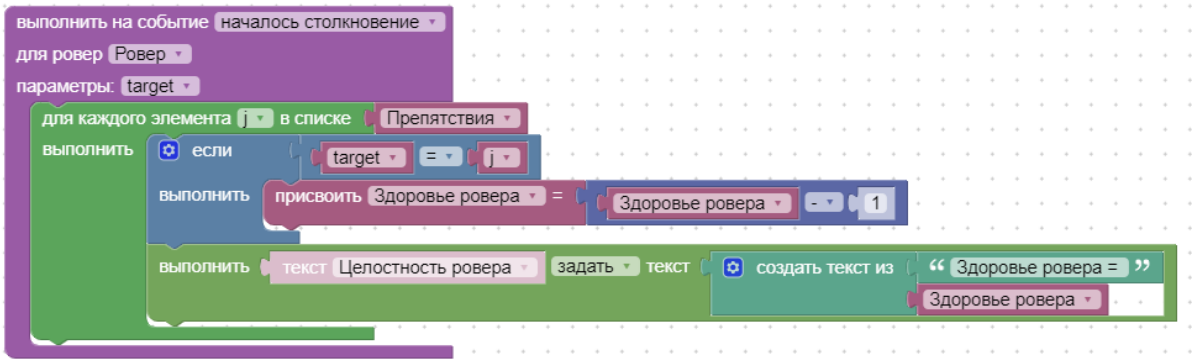
- Реализовать интерфейс и логику управления ровером
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с логическими блоками в Blockly
- Закрепить навыки работы с активацией/деактивацией объектов
- Закрепить навыки работы с продвинутыми функциями текста
- Закрепить навык тестирования своих проектов

- Закрепить навыки работы с UI/UX - дизайном
- Закрепить навыки работы с событиями “Объект начали использовать”

Вкладка Логика столкновений

Сейчас мы приступим к одному из самых трагичных моментов нашей логики приложения - к столкновению ровера с препятствиями, которое может привести к его полному разрушению. С другой стороны, когда ровер столкнется с Маяком - это хорошо, тогда мы перейдем к следующему этапу Космической миссии.

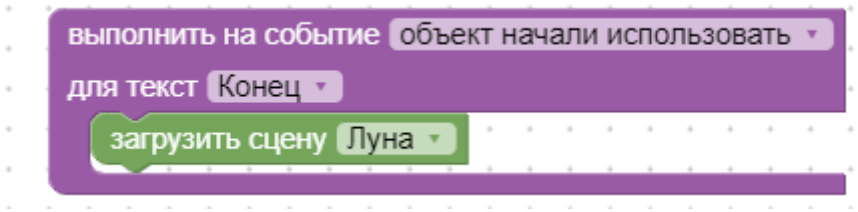
Для реализации логики столкновений снова обратимся к Циклам и перебору элементов их Списка Препятствий:



Логика здесь следующая: в момент столкновения *Ровера* с каким-нибудь объектом цикл перебирает объекты из списка *Препятствия*, если один из объектов является target (целевым), т.е. соответствует объекту из списка *Препятствия*, то переменная *Здоровье ровера* уменьшается на одну единицу (декремент) и обновляется информация на UI “*Целостность ровера*”, которое мы разместили на сцене в предыдущем пункте (п. 8.2.3.).

Рассмотрим самый худший вариант, когда мы снизили здоровье ровера до “0”:

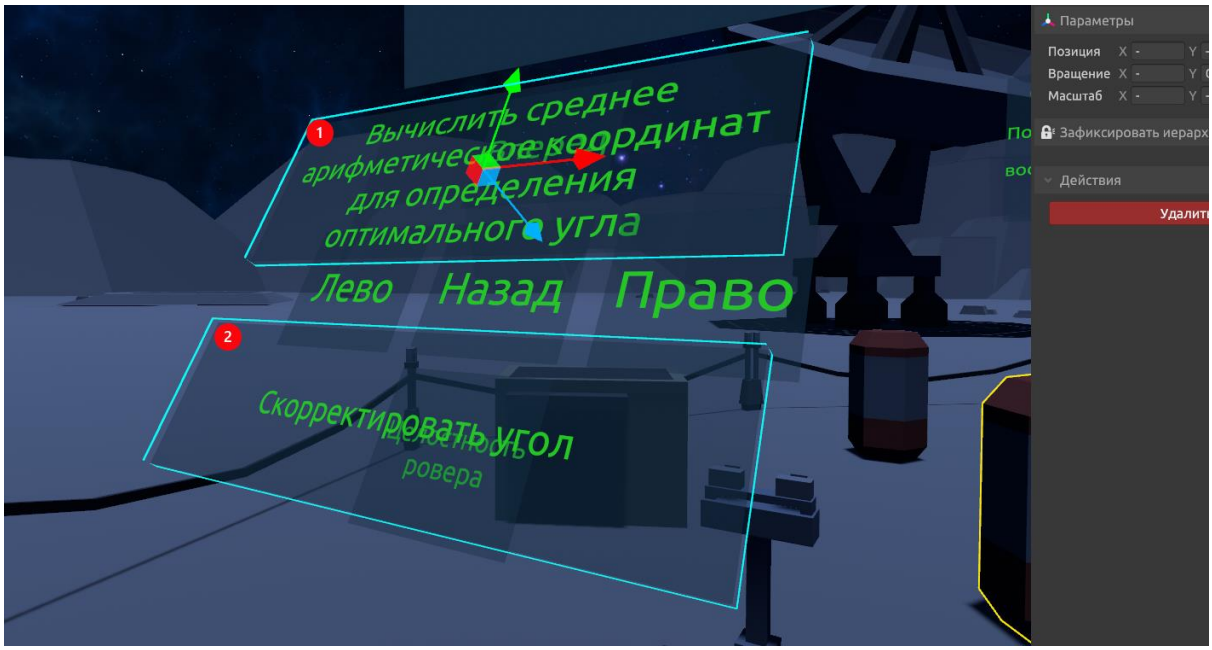
В этом блоке мы активируем UI “Конец” и устанавливаем на нем текст: “Ровер уничтожен. Миссия не может быть выполнена. Нажмите на эту панель для перезапуска”.



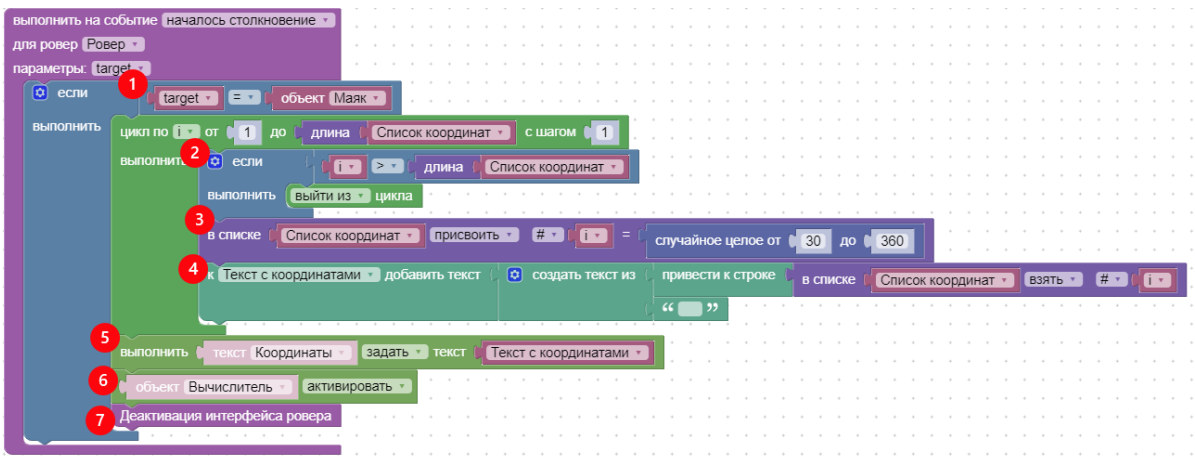
Блок выше отвечает за перезапуск сцены. Далее он нам пригодится и для счастливого финала, только текст будет иной.

Для активации счастливого финала *Ровер* должен столкнуться с *Маяком*.

Перед созданием логики столкновения *Ровера* с *Маяком*, давайте разместим на локации поверх интерфейса *Ровера* еще два объекта UI, которые будут отвечать за управление вычислением Оптимального угла (UI “***Вычислитель***”) и поворотом *Радара* на Оптимальный угол (UI “***Корректировщик***”):



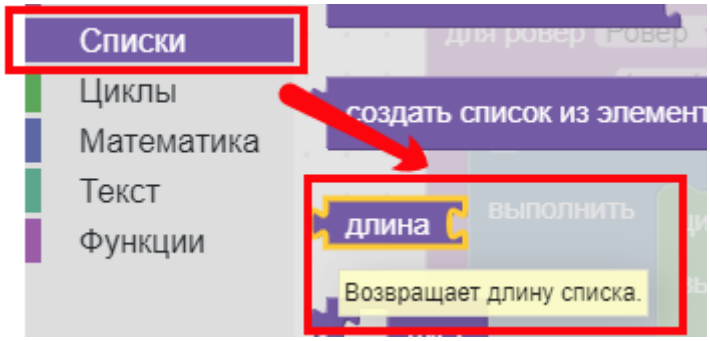
Теперь разберем как работает блок события при столкновении с *Маяком*:



Логика здесь следующая: в момент, когда *Ровер* сталкивается с объектом и если этот объект является *Маяком* запускается ***Цикл по i*** **(1).**

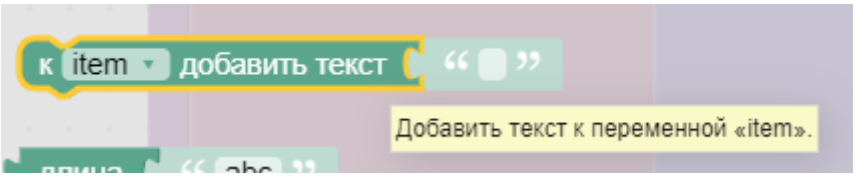
Данный цикл перебирает и создает случайные значения для каждого элемента в списке *Список координат* (от 1 до “Длины списка “Список координат”, **2**).

Блок Длина списка находится в категории “Списки”:



Примечание*: Вы помните, что в Момент инициализации мы создавали список Координат из 5 элементов, но с помощью блока Длина списка мы можем автоматически измерить длину этого списка (иначе нам каждый раз вручную пришлось бы вводить количество элементов при изменении их количества).*

Пока *i* меньше длины списка мы перебираем его элементы, и для каждого элемента присваивается случайное значение угла от 0 до 360 **(3)**, а также мы дополняем переменную ***Текст с координатами*** этим сгенерированным значением.
Для этого используем блок из Категории “Текст” **(4)**:



Таким образом, с каждой новой итерацией цикла мы добавляем к переменной ***Текст с координатами*** новое значение, которое будет не больше *Длины Списка Координат*. В нашем случае цикл повторится 5 раз.

Как только *i* будет больше пяти (*длины списка Координат*), то выполнится команда ***Выйти из цикла.***
На этом Цикл будет завершен.

Последнее, что нам нужно будет сделать - это активировать объект UI **“*Вычислитель*”**, который будет отвечать за вычисление *Оптимального угла* и активировать функцию ***Деактивация интерфейсов ровера*** в теле условного оператора, поскольку далее по логике сценария эти интерфейсы нам больше не понадобятся.

И уже на следующем занятии мы перейдем к самой приятной финальной части.

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. В каких случаях нам может пригодиться длина списка?
2. Что такое target и для чего он нужен?

Занятие 7.7 Настройка математической логики проекта

Цель:

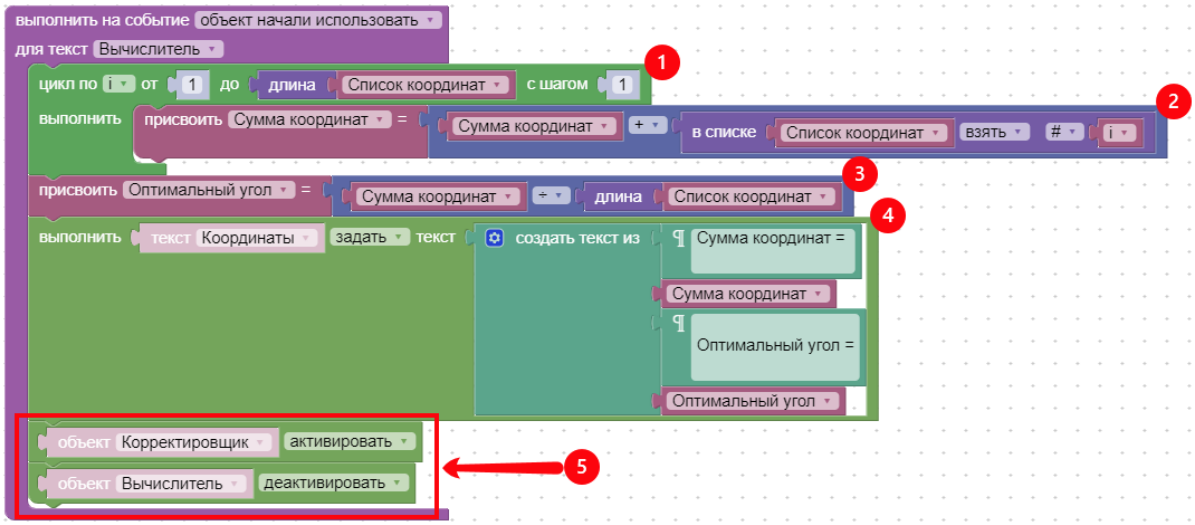
Разработать для проекта "Космическая миссия" вкладку логики “Установление контакта и финал миссии”. Завершить разработку проекта "Космическая миссия".

Задачи:

- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с логическими блоками в Blockly
- Закрепить навыки работы с активацией/деактивацией объектов
- Закрепить навык тестирования своих проектов
- Закрепить навыки работы с UI/UX - дизайном
- Закрепить навыки работы с событиями
- Усвоить навыки работы с математическими блоками
- Закрепить навыки работы с циклами

Вкладка Установление контакта и финал миссии

Во первых, разберем логику события при использовании UI “*Вычислитель*”:



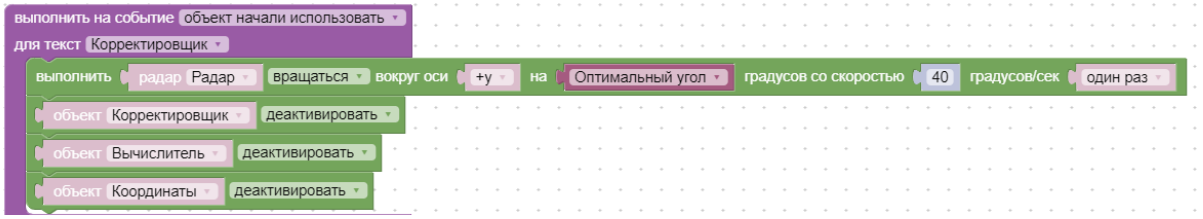
Логика здесь следующая:

В момент, когда мы нажимаем на UI “Вычислить среднее арифметическое координат для определения Оптимального угла” (UI “*Вычислитель*”) начинает

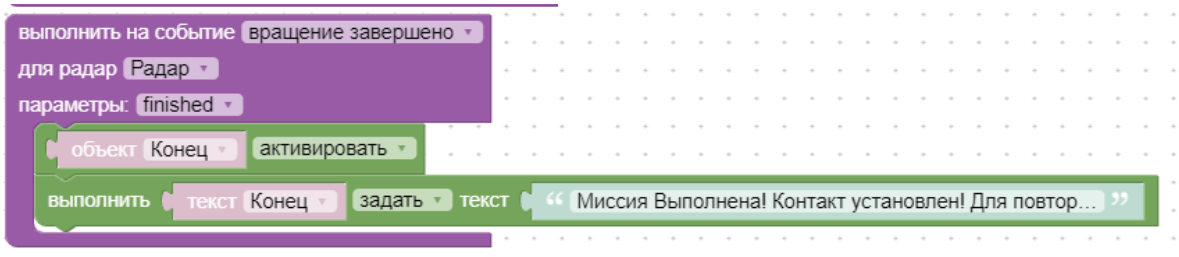
работать **Цикл по i (1)**, который перебирает все элементы из списка *Сумма координат*, который ранее был сгенерирован случайными числами от 0 до 360.

Мы суммируем все сгенерированные координаты - в каждой итерации цикла к переменной *Сумма координат* прибавляется значение по индексу **(2)**.
После этого автоматически вычисляется переменная *Оптимальный угол* по методу **среднего арифметического (3)**. Сумма координат делится на длину Списка координат (в нашем случае эта длина равна 5).

***Примечание**: Среднее арифметическое (в математике и статистике) — разновидность среднего значения. Определяется как число, равное сумме всех чисел множества, деленной на их количество.*
После этого мы задаем текст в UI, отвечающее за сообщение от *Маяка* (UI "*Координаты*"), информирующее Игрока о *Сумме координат* и *Оптимальном угле* **(4)**.
Последнее, что мы делаем - это деактивируем UI "*Вычислитель*", поскольку по сценарию далее это UI нам не понадобится и активируем последнее UI "*Скорректировать угол*" (UI "***Корректировщик***"), отвечающее за поворот Радара на *Оптимальный угол* **(5).**
Далее мы должны использовать UI "*Корректировщик*", чтобы запустить следующую цепь событий:

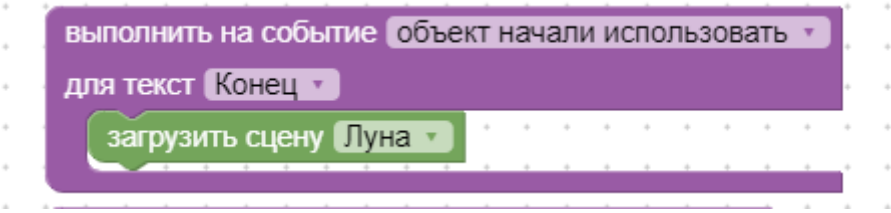


Радар вращается на количество градусов равное значению Оптимального угла, чтобы выполнить требование из ТЗ: *Радар должен поворачиваться на вычисленный оптимальный угол, который задается случайным образом при каждом запуске приложения.*
В этот же момент мы деактивируем оставшиеся UI для того, чтобы нам ничто не мешало лицезреть последний логический аккорд нашей Космической миссии:



Именно так выглядит Наш счастливый финал, задаем UI "*Конец*" следующий текст: *"Миссия Выполнена! Контакт установлен! Для повторения миссии нажмите на эту панель!"*.

Вы помните, что еще во вкладке **Логика столкновений** мы задавали этому UI “*Конец*” следующую логическую конструкцию и она будет работать и в нашем счастливом случае:



Если вы все сделали правильно, то в финале вы должны увидеть следующую сцену:

Контрольные вопросы (выборочно можно использовать на этапе рефлексии, для проверки усвоения знаний, полученных на занятии):

1. Опишите логику работы логического блока цикла "по i".
2. Как вывести переменную в текст?
3. Какие свойства настраиваются в продвинутом действии вращения?

Занятие 7.8 Свой проект

Цель:

Усвоение навыков, полученных в ходе практических занятий. Учитывая полученный опыт в ходе обучения по данной программе финальным заданием программы обучения является выполнение креативного проекта по собственному техническому заданию.

Задачи:

- Закрепить навык тестирования работоспособности собственных проектов
- Повысить навыки исправления багов/ошибок в проекте
- Повысить навыки рисования скетчей/ планов перемещения по виртуальному пространству

- Закрепить навыки создания и использования дизайна интерфейсов, основанном на удобстве для конечного пользователя
- Усвоить навыки работы с функциями в Varwin
- Усвоить навыки работы с таймером
- Усвоить навыки работы с расширенным функционалом текста

Формат: Самостоятельная работа, решение кейсового задания.

Оборудование: несколько листов бумаги А4 на каждого обучающегося, комплект карандашей.

0 - 10 минут	Получение кейса. Формирование собственного технического задания по проекту "Урок английского".

На этом этапе обязательно нужно сформировать и зафиксировать техническое задание на проект

Шаблон технического задания:

1. Тематика/направленность проекта;
2. Сколько и каких объектов будет использовано (обязательно указывать пакеты объектов, которые будут использованы или же свои объекты);
3. Основной функционал приложения;
4. Дополнительные возможности приложения;
5. План-схема сцен, которые будут реализованы;
6. Каким образом будет реализован, удобный для пользователя, UX/UI - дизайн;
7. Какую полезную функцию для общества несет проект, или будет нести при дальнейшей его доработке.

10 - 40 минут	Разработка проекта согласно техническому заданию.

Применяя ранее полученные навыки, обучающиеся разрабатывают проекты и тестируют их на VR-HMD устройствах.

40 - 45 минут	Сохранение проектов. Тестирование и демонстрация проектов на VR-HMD устройствах.

Рекомендация: если есть возможность, то можно посвятить презентации проектов отдельное занятие. Для повышения soft-skills и обмена опытом между обучающимися.

Кейс:

Финальный проект выполняется под контролем наставника и его помощью с точки зрения концепции и позиционирования проекта. С технической части наставник никаким образом не помогает в реализации.

Самый главный момент в финальном кейсе это генерация идей. Её нужно провести совместно всей группой. Желательно использовать популярные инструменты, такие как, *мозговой штурм* или *метод 6 шляп.*

После выбора определенной максимально жизнеспособной идеи обучающие составляют и фиксируют техническое задание. После этого начинается такая же разработка, как и при сборке кейсового задания.

Обучающимся предоставляется творческая свобода и они сами решают какой проект они будут разрабатывать и как именно они будут это делать. Соответственно, о ответственность за проект повышается.

Обязательные условия:

1. Сформировать и зафиксировать технического задание проекта
2. Нарисовать план расположения объектов на сценах
3. Зафиксировать дополнительные функции, которые будут реализованы в проекте