3D Action Template – это простой шаблон для создания экшен (или квест) игр от первого лица, основанный на C# компонентах.

Содержит:
- Меню на основе 3D объектов в сцене
- Понятия «игра» и «миссия» (делится на подзадачи)
- HUD
- Контроллер управления от первого лица (бег, прыжки, приседания)
- Инвентарь, взятие и использование предметов и оружия
- Оружие (ближнего и дальнего боя)
- Автомобили и самолеты, которыми может управлять игрок
- ИИ враги, которые умеют патрулировать и атаковать

Установка: перед импортированием пакета удалите папки **bake_lighting** и **csharp_template** вашего C# проекта, чтобы исключить коллизии файлов (иначе освещение будет поломано, а персонаж будет двигаться не так, как задумано)!

Для запуска демо игры, выберите в редакторе лаунчер «**RunDemo**», либо откройте мир: “data/AlexanderPanichev/3DActionTemplate/sample/worlds/main_menu.world” и нажмите «Play».

Для запуска упрощенного семпла, выберите в редакторе лаунчер «**RunSample**» (либо откройте мир “.../worlds/sample.world”).

Архитектура

Взаимодействие между компонентами реализуется через **события** (**System.Action**). Одни компоненты события **создают**, а другие **принимают**.

На примерах это выглядит так:
1) “**Кнопка**” и “**Выход из игры**”. Первое создает событие (срабатывает при нажатии), а второй событие принимает (некий класс, который умеет закрывать игру).
2) “**Игрок зашел в комнату**” и “**Создание врагов**”. Первое триггер, создает события. Второй принимает (триггерится) и создает врагов.
3) “**Ручка двери**” и “**Открытие двери**” — ручка двери создает событие при попытке провернуть ее. Дверь открывается, если на ней есть компонента, принимающая это событие (если кто-то настроил это в редакторе). Благодаря такому подходу, дверь может открываться при любых типах событий: например, можно настроить, чтобы дверь открывалась сама, если игрок «зашел в комнату», либо «убил всех врагов в комнате», либо «нажал на кнопку в другом углу комнаты» и т.д.
4) “**Здоровье игрока**” и “**Game Over**”. Первое может генерировать разные виды событий: «когда здоровье кончились», «когда здоровье полностью восстановились», «когда здоровья меньше половины» и т. д. В данном случае «Game Over» подписывается на событие «когда жизни кончились».

В коде это выглядит так: есть класс **CEventHandler**, от которого **наследуются все компоненты**, которые могут **принимать** события. Для обработки входящих событий используется виртуальный метод **Activate**():

CEventHandler.cs

data > AlexanderPanichev > 3DActionTemplate > template > components > common > CEventHandler.

```
using System;
using System.Collections;
using System.Collections.Generic;
using Unigine;

[Component(PropertyGuid = "60f410ec0e9dd75be62424feefd008451c07dce0")]
public class CEventHandler : Component
{
    public Action onActivated;

    public virtual void Activate(Component sender) {}

    public void OnReceiveEvent(Component sender)
    {
        Activate(sender); // notify derived class
        onActivated?.Invoke(); // notify subscribers
    }
}
```

Компоненты, которые **создают** события, используют **публичные** списки **List<CEventHandler>** (чтобы можно было назначать получателей прямо в редакторе):

CPhysicalTrigger.cs

data > AlexanderPanichev > 3DActionTemplate > template > components > quest > CPhysicalTrig

```
using Unigine;

[Component(PropertyGuid = "957f398422b3771321d891566fbc98bdf85302f8")]
public class CPhysicalTrigger : Component
{
    public enum Mode
    {
        Single,
        Multiple,
    }
    public Mode activation_mode;
    public List<CEventHandler> onEnter = new List<CEventHandler>();
```

А при наступлении события сообщают о нем всем назначенным получателям:

```
// notify subscribers
foreach (var receiver in onEnter)
    receiver.OnReceiveEvent(this);
```

Все компоненты шаблона лежат тут:
data/AlexanderPanichev/3DActionTemplate/template/components/

Поделены на 5 подпапок:

1) **common** – общие базовые компоненты игрока (управление игроком, HUD, игра)
2) **main_menu** – компоненты для создания меню (кнопки)
3) **quest** – компоненты для создания квест игр (двери, предметы, триггеры)
4) **shooter** – компоненты для создания шутера (оружие и враги)
5) **vehicles** – компоненты для созания техники (машины, самолеты, экзоскелеты)

Компоненты common

CEventHandler – класс, от которого нужно наследоваться и переопределять метод Activate(), чтобы уметь получать события. В демо эта компонента используется в качестве индикатора того, что игрок что-то выполнил (смотрите ноду game и компоненту CGame.cs).

CGame – синглтон, содержит ссылки на игрока, его здоровье, список задач для выполнения миссии.

CHUD – минималистичный интерфейс игрока. Показывает текущую цель, подсказки по управлению, информацию о здоровье игрока.

CPlayer - «игрок», а точнее информация о нем. Содержит ссылки на камеру, место крепления для оружия/предметов, текущее здоровье и инвентарь. Этот же класс позволяет игроку взаимодействовать с миром: брать предметы и использовать их на других предметах. Где-то внутри иерархии должен содержать компоненту CHealth.

FirstPersonController – управление игроком. Хождение, бег, прыжки, приседание, управление камерой.

UI – класс-обертка над движковыми виджетами, добавляющее к ним pivot, anchor и автомасштабирование интерфейса в зависимости от текущего разрешения. **Крайне рекомендую использовать Toolkit аддон (ищите его в Unigine Add-on Store) вместо этого класса. Toolkit является улучшением данного класса и имеет отдельный визуальный редактор интерфейса.**

Компоненты main_menu

CChangeCamera – меняет текущую камеру при получении события.

CRunConsoleCommand – запускает консольную команду при получении события (например, может загрузить другой мир)

CUIButton – компонент, который накидывается на объект класса Unigine.Object (ObjectMeshStatic, например). Кнопка, которая генерирует события при нажатии на нее.

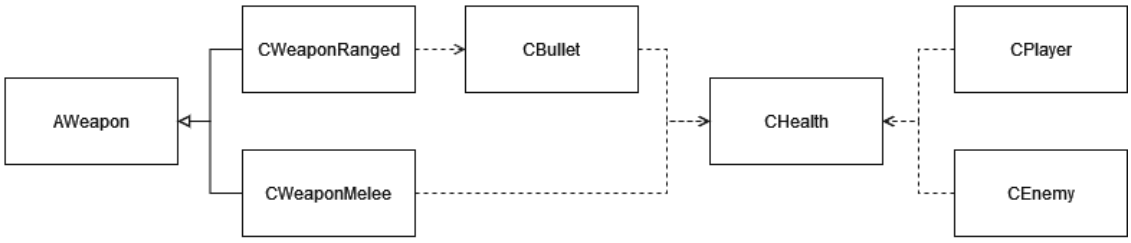
Компоненты quest

CDoor – дверь, которая может открываться или закрываться при срабатывании события.

CInteractable – интерактивный объект, который можно взять, использовать, либо который требует какой-то объект в руках игрока, чтобы была возможность использовать его. Вызывает событие OnInteract. Чтобы предмет можно было использовать, он должен быть типа **Unigine.ObjectMeshStatic**! Также, **Item Mask** должен совпадать с **Item Mask у CPlayer**'а, чтобы можно было взаимодействовать с этим объектом.

CPhysicalTrigger – триггер, который при вхождении игрока создает событие OnEnter. Должен назначаться на объект типа **Unigine.PhysicalTrigger**!

Компоненты shooter



AWeapon – абстрактный класс оружия. От него наслеуются CWeaponMelee и CWeaponRanged.

CBullet – компонента, которая накидывается на объект-пулю (снаряд).

CDestroyTimer – компонента, которая уничтожает объект, на котором висит, через «timer» секунд.

CEnemy – враг. Умеет патрулировать, атаковать и искать (стоять на месте). Компонента должна назначаться на объект типа **Unigine.Object** с добавленным **Body (Rigid)**!

CHealth – компонента, которая вешается как на игрока, так и на врагов. Управляет здоровьем юнита. Все виды оружия взаимодействуют только с этой компонентой.

CKillEnemiesTask – компонента, которая вызывает событие OnKillAll, когда все враги из списка Enemies будут мертвы.

CWeaponMelee – оружие ближнего боя.

CWeaponRanged – оружие дальнего боя.

Компоненты vehicles

CAirplane – контроллер управления самолетом. Назначается на объект типа **Unigine.Object** с добавленным **Body (Rigid)**!

CCar – контроллер управления автомобилем. Назначается на объект типа **Unigine.Object** с добавленным **Body (Rigid)**!

CKeyPressed – триггер, который вызывает событие OnPressed при нажатии клавиши на клавиатуре.

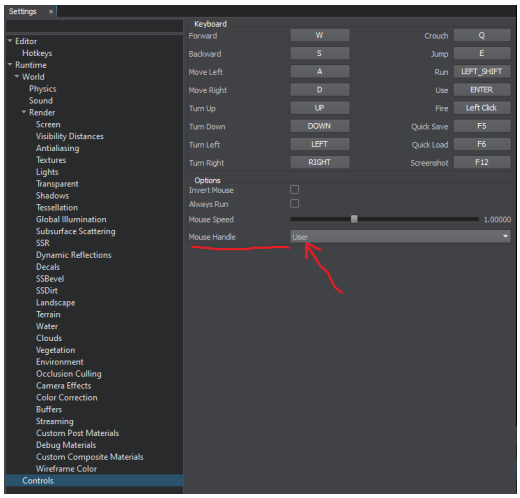
CSwitchPlayer – компонента, которая по событию переключает разные виды игрока.

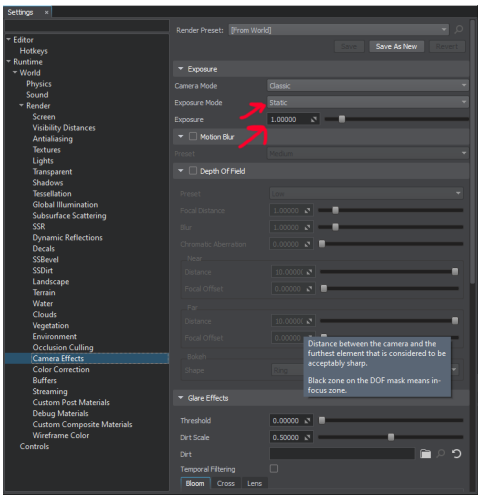
Как работает главное меню

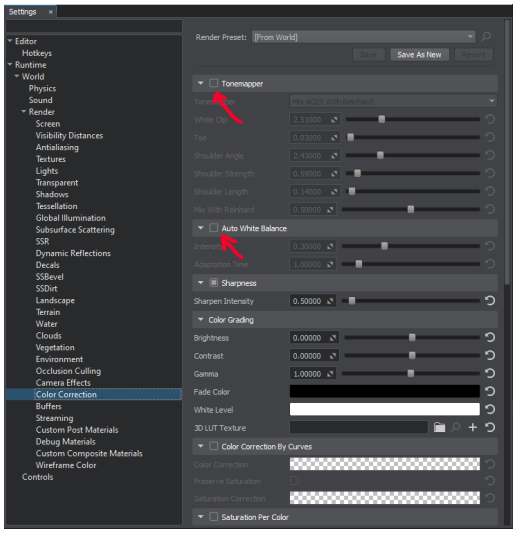
Я крайне рекомендую использовать **Toolkit** аддон из Unigine Asset Store для создания интерфейсов: https://store.unigine.com/add-on/1f05cd6d-db59-6eee-a4f7-9ba129dedac0/description

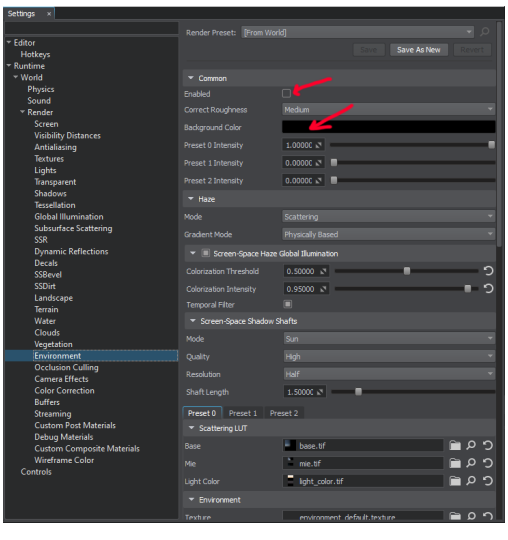
Здесь же показан упрощенный пример, основанный на объектах сцены. Кнопки — это объекты ObjectMeshStatic, висящие перед «глазами» камеры. В каждом кадре выпускаются лучи из камеры через курсор в мир. Если есть пересечение с объектом, то этот объект считается выделенным.

Для начала надо настроить сцену. Выключаем всё окружение и переводим курсор в режим «постоянно виден и не захватывается никем» (MouseHandle.User):

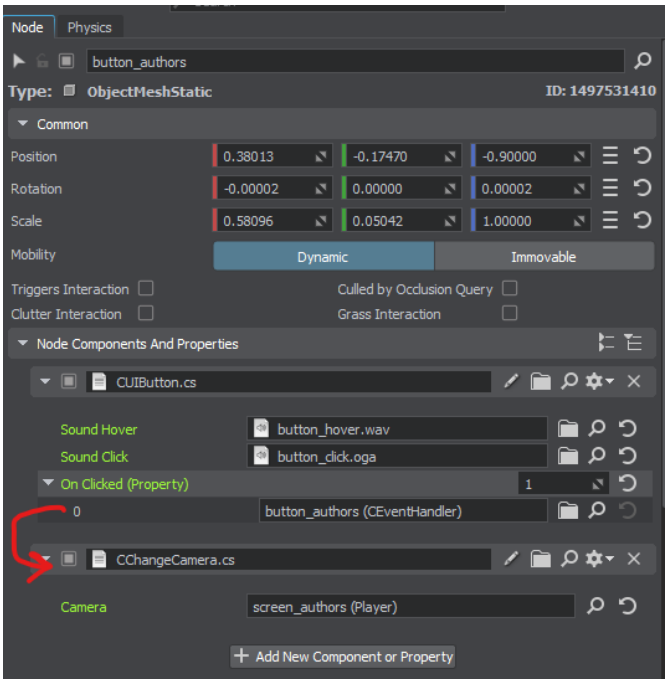








Далее рассмотрим, как выглядит кнопка, которая меняет страницу (на самом деле происходит переключение на другую камеру).

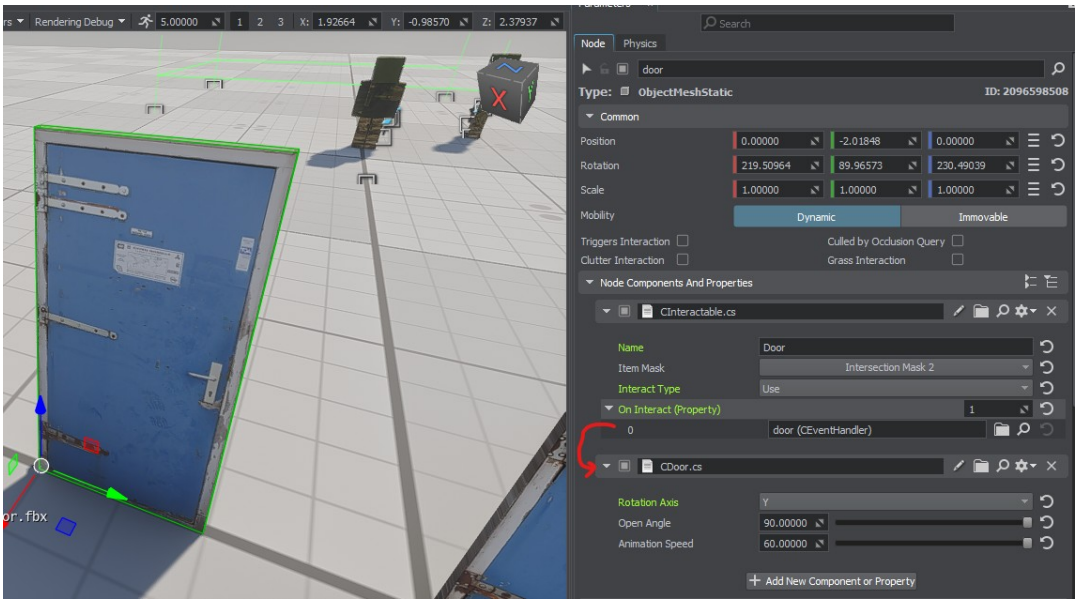


Это ObjectMeshStatic в виде плоскости, повернутой к камере. Здесь есть две компоненты: **CUIButton** и **CChangeCamera**. Первая при наведении мышкой на объект и нажатии на левую кнопку мыши вызывает событие OnClicked, которое передается единственному подписчику — компоненте CChangeCamera, которая висит на той же ноде. Она, в свою очередь, просто переключает текущую камеру на другую.

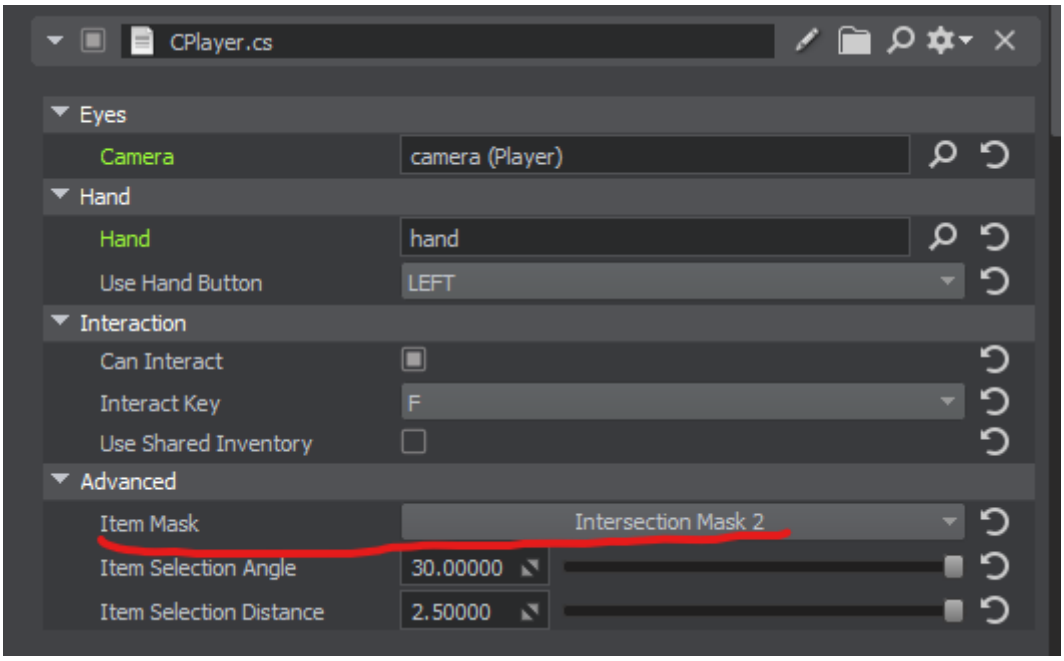
С таким же подходом работают все остальные кнопки интерфейса.

Как открываются двери

Дверь — это интерактивный объект, который можно использовать игроком. Логика похожа на то, что мы видели в главном меню:



Есть две компоненты: **CInteractable** и **CDoor**. Первая говорит о том, что этот объект можно **поднять** или **использовать** игроком. Чтобы можно было использовать дверь, убедитесь, что Item Mask совпадает с тем, что написано у CPlayer'а (самого игрока):



А также то, что у «**Can Interact**» стоит галочка. Она включает логику взаимодействия с предметами (с компонентами CInteractable).

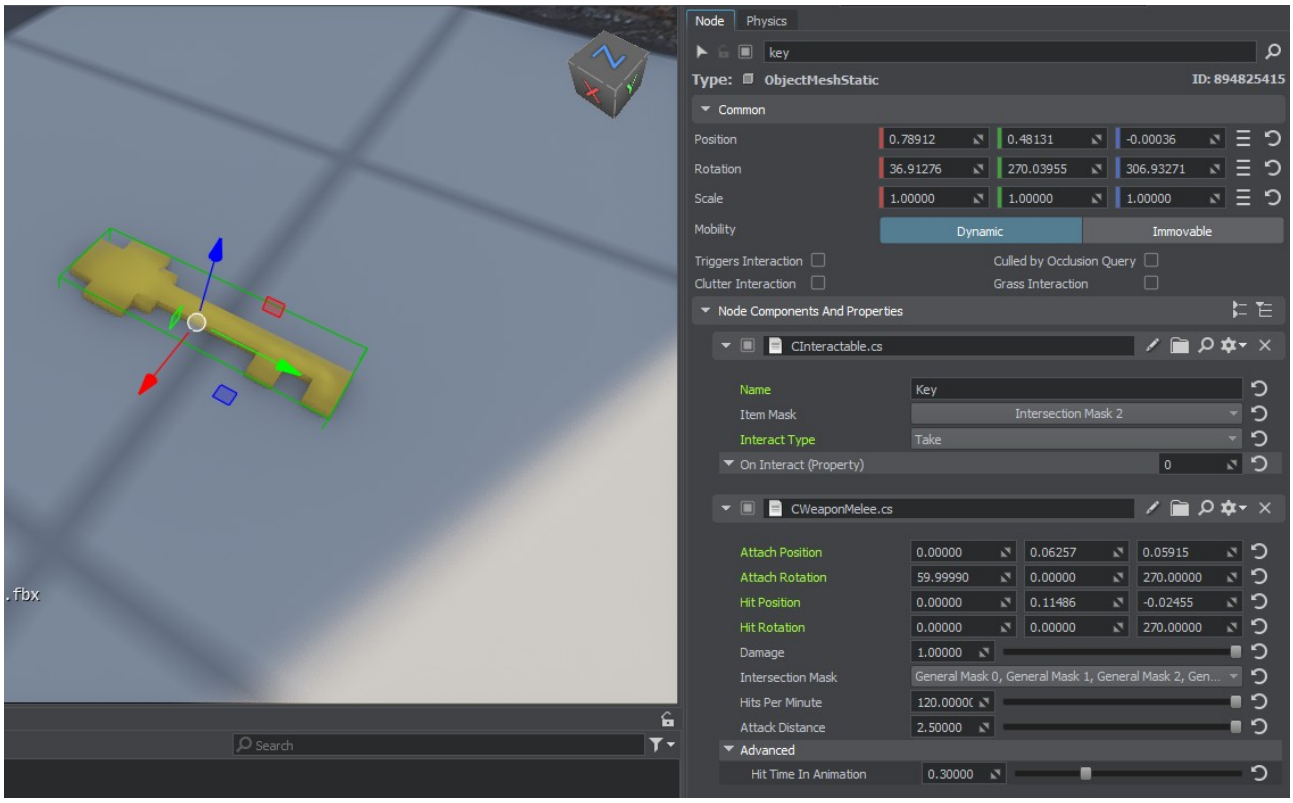
Item Selection Angle – под каким углом можно смотреть на объект, чтобы с ним можно было взаимодействовать.

Item Selection Distance – на каком расстоянии максимум можно быть, чтобы иметь возможность взаимодействовать с объектами.

Так вот, вернемся к двери. При нажатии на F, дверь вызовет событие OnInteract и сообщит об этом подписчику — компоненте CDoor, которая отвечает за плавную анимацию открытия и закрытия двери.

Как работает оружие

Рассмотрим ключ, который в семпле является еще и оружием ближнего боя.



Здесь есть две компоненты: **CInteractable** и **CWeaponMelee**. Первая говорит о том, что ключ (оружие) можно подобрать (Interact Type == Take) и поместить в инвентарь игроку, если Item Mask у предмета и игрока (CPlayer) будут совпадать.

Вторая компонента — это описание самого оружия.

Attach Position/Rotation – куда это оружие должно встать относительно руки игрока (hand нода) при выборе оружия путем скролла мыши.

Hit Position/Rotation – куда оружие должно наклониться при ударе.

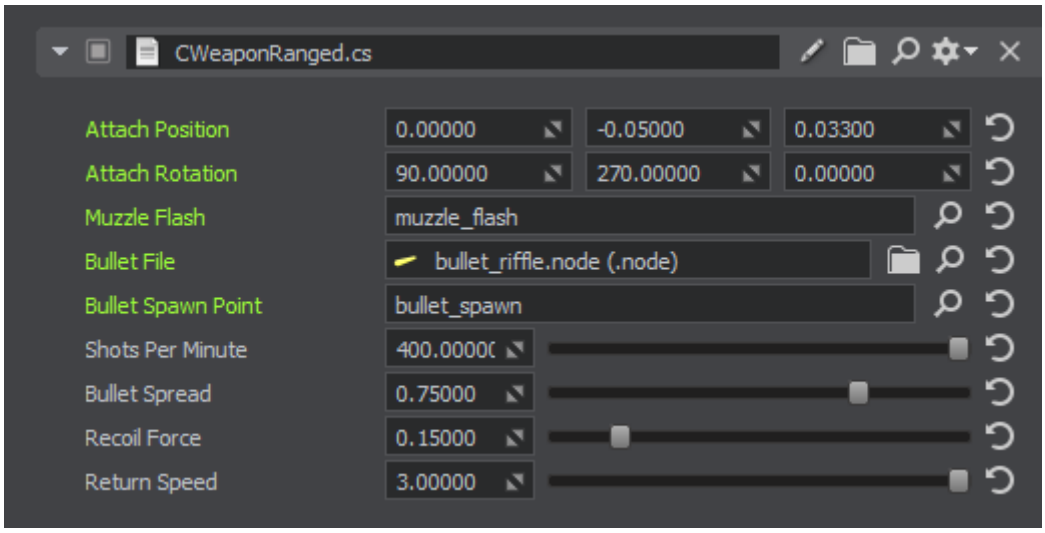
Damage – сколько урона наносит.

Intersection Mask – каким видам врагов наносит урон (маска должна совпадать на сюрфейсах врагов, чтобы урон учитывался)

Hits Per Minute – ударов в минуту

Attack Distance – максимальное расстояние для удара.

Оружие дальнего боя (пистолеты, автоматы, луки) использует компоненту **CWeaponRanged**:



Muzzle Flash – вспышка при выстреле.

Bullet File – нода вылетающего патрона (ядра). Внутри должна иметь компоненту CBullet.

Bullet Spawn Point – откуда вылетает патрон (ядро)

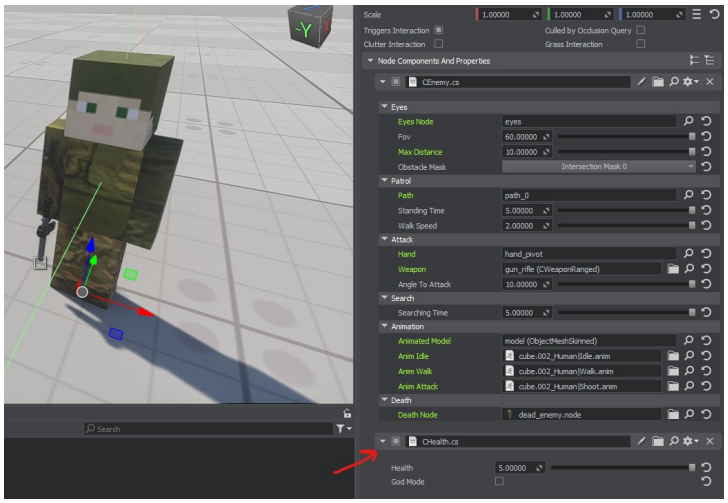
Shots Per Minute – выстрелов в минуту

Bullet Spread – разброс выстрелов

Recoil Force – сила отдачи (только анимация)

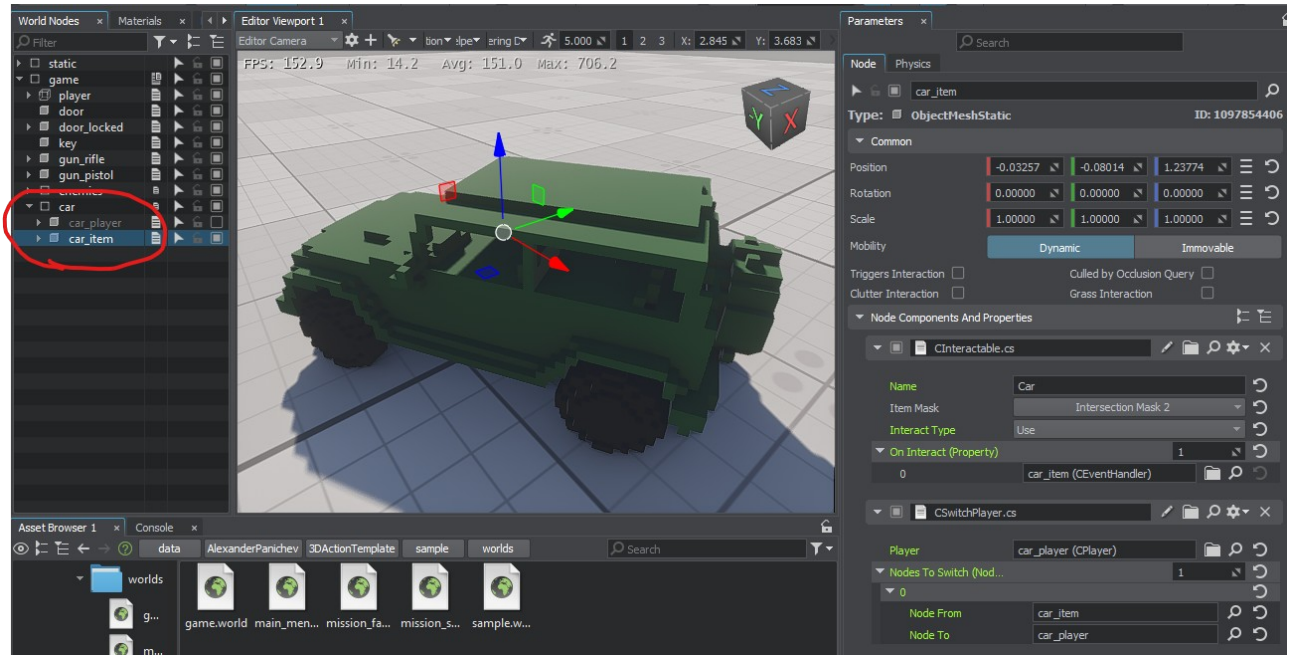
Return Speed – скорость возвращения оружия в дефолтное положение после выстрела (анимация)

Чтобы оружие наносило урон, у врага должна быть компонента **CHealth**:



Как работает техника

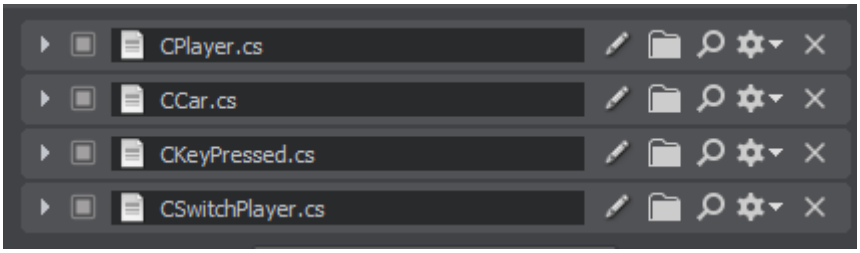
Техника — это просто разные виды игроков, которые вы переключаете во время игры. На сцене это выглядит так:



В данном примере есть две ноды:

car_item – машина как «предмет». Содержит всего 2 компоненты: **CInteractable** – чтобы можно было «сесть в машину». **CSwitchPlayer** – компонента, которая переключает игрока (выключает управление текущим, а также активирует car_player — игрока, и выключает car_item – машину как «предмет»).

car_player – машина как «игрок». Активен, когда игрок сидит в машине. Содержит 4 компоненты:



CPlayer – информация о машине в качестве «игрока»: может ли она «брать» предметы, где у нее камера и т. д.

CCar – характеристики авто и его управление. Можно сказать, что это аналог FirstPersonController, но для управления транспортом.

CKeyPressed – триггер, который срабатывает при нажатии кнопки (вызывает событие OnPressed). В данном случае нужен, чтобы из машины в любой момент можно было выйти по нажатию на клавишу F. Подписчик события — компонента **CSwitchPlayer**, которая выключает car_player и включает car_item, возвращая игрока как «пешехода».