Antares Browser & Addon (1.5.0)

(Open Source project)

http://antarespb.sourceforge.net/

1. Описание.
2. Установка.
3. Features

ОПИСАНИЕ.

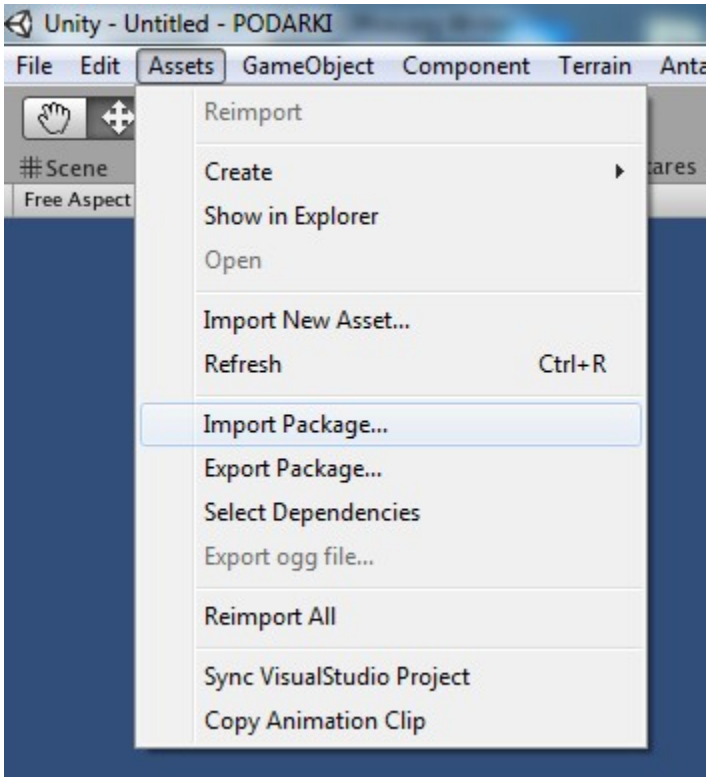
Antares Browser & Addon это пакет C# классов, расширяющих функционал и возможности Редактора Unity, а так же, добавляющие некоторые дополнительные методы для улучшения управления вашим проектом и его кодом.

Antares Browser & Addon является Open Source проектом и вы можете использовать его бесплатно в любом колличестве проектов.

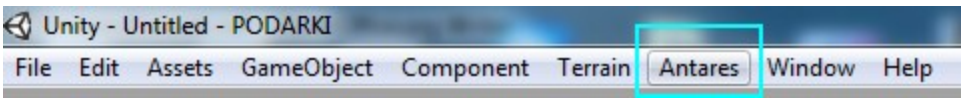
УСТАНОВКА.

Самую новую версию Antares Browser вы можете найти здесь : http://antarespb.sourceforge.net/

- Скачайте **AntaresBrowserAndAddon**(номер версии)**.unitypackage**
- Запустите Редактор **Unity**.
- Выбирете **Assets → Import Package**



- Наведите курсор на любую команду в верхней строке. На пример — на **Assets.** Окно Редактора обновится и вы увидите новый элемент **Antares**



- Поздравляю! **Antares Browser and Addon** установлен в ваш проект.

FEATURES.

Броузер :

1. Броузер Скриптов.
2. Броузер Моделей.
3. Броузер - Текстуры

Аддон :

4. **Компонент Antares Addon**
5. **Настройки AntaresAddon**
6. Инструменты для Редактирования Уровней (Level Design)Level Design)

7. **Кривые Безье (Bezier Curves)**
    1. Создание Кривых в Редакторе.
    2. Редактирование Кривой в Редакторе Unity.
    3. Создание проводов (wires)
    4. Спирали
        1. Настройки Спиралей
    5. Динамическое создание и редактирование Кривых.
    6. Curve API
    7. Known bugs.

8. **Генератор дорог на базе Кривых (Curved Road)**
    1. Создание дороги.
    2. Шейдеры.

9. **1.Система Мультитагов (Multitags)**
    1. Multitags API

10. **1.Модуль для сохранения описания объектов.**

11. **ANTARES : BASKET**
    1. **Экспорт в формат Asset Bundle**

12. **ANTARES : DUBLICATOR**

13. **Визуализация Джоинтов (Joints)**

14. Инструменты : Biped

15. Инструменты : Game Object
    1. INTERPOLATOR

16. Инструменты : Camera

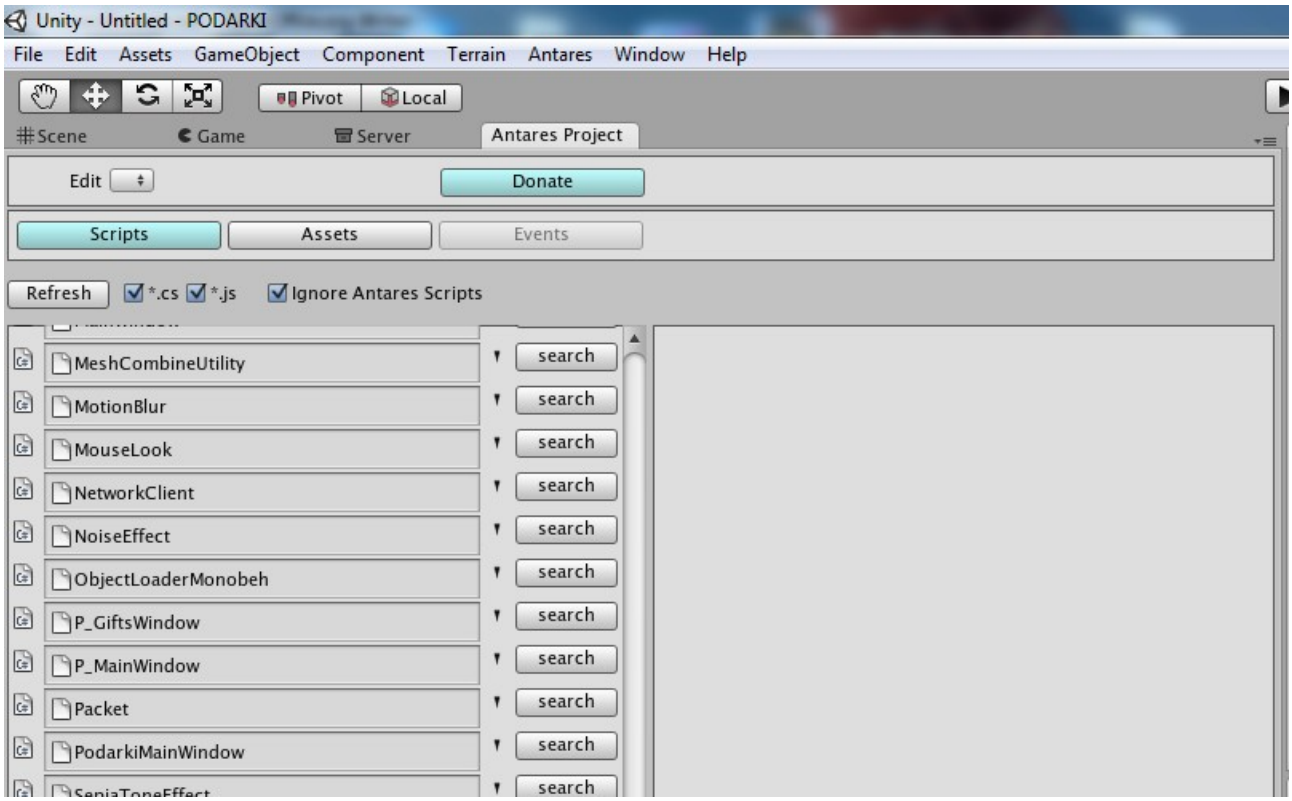
17. **ANTARES.dll (условно бесплатно) :**
    1. **ANTARES.dll : Manager**

2. **ANTARES.dll : AGUI**
3. **ANTARES.dll : BinarySave**
4. **ANTARES.dll : Async**

18. **ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ**
    1. Align with view
    2. Copy Animation Clip
    3. Hide / Unhide в окне Иерархии

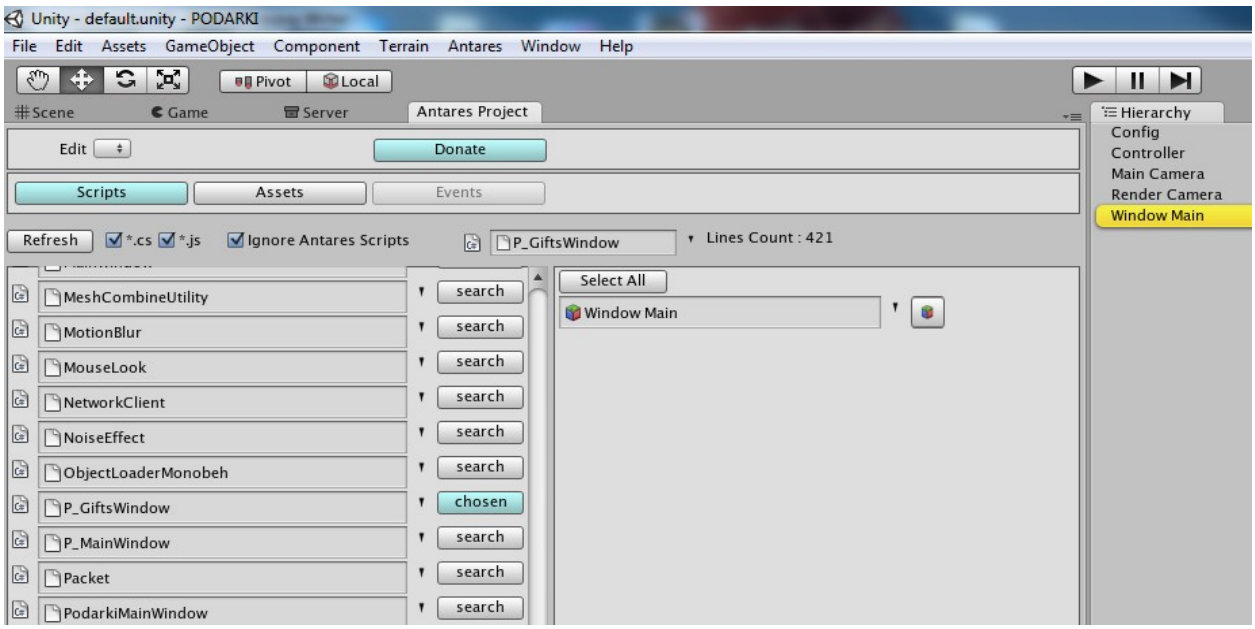
19. Авторы
20. **Помочь Проекту.**

БРОУЗЕР СКРИПТОВ.

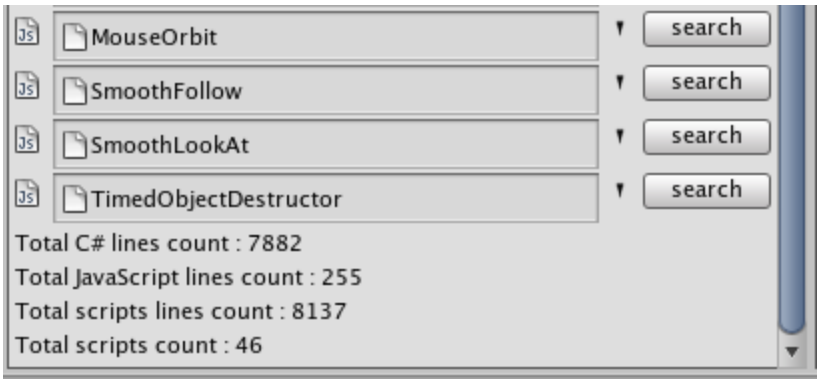


Броузер скриптов ищет и отображает список всех C# и JavaScript файлов проекта. Boo на данный момент не поддержан.

При нажатии кнопки «Search», производится поиск всех объектов в Сцене, использующих этот скрипт. Если объекты найдены, они отображаются списком в правой части окна Броузера.



В нижней части списка скриптов, отображается статистическая информация :



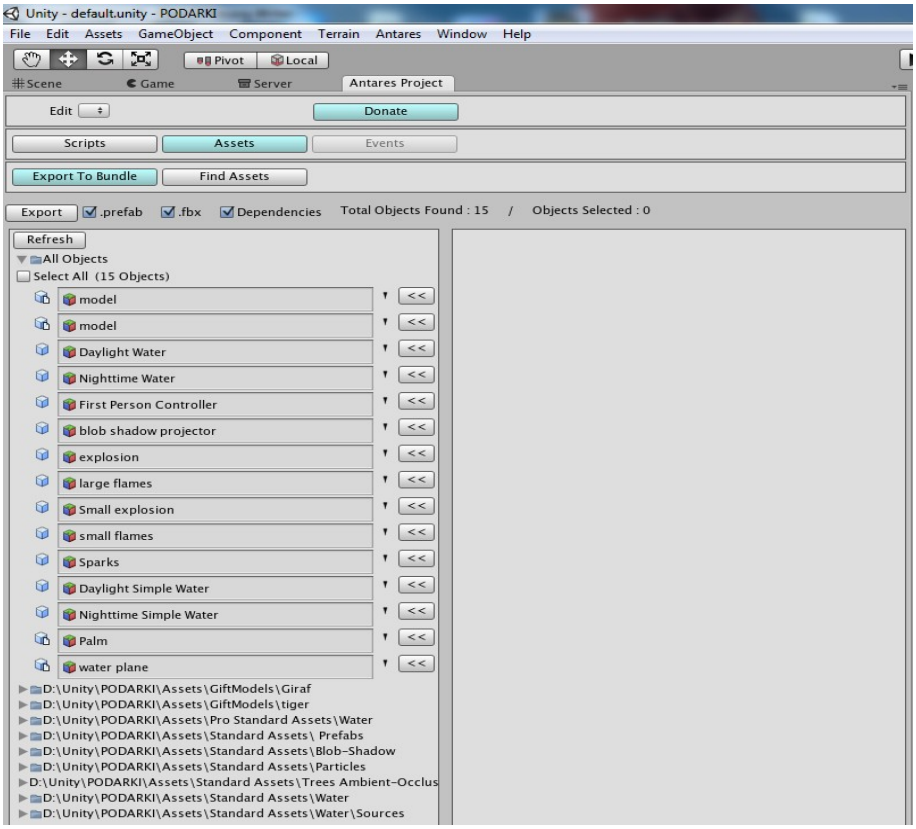
БРОУЗЕР МОДЕЛЕЙ

Так же, как в Броузере Скриптов ищет скрипты, Броузер Моделей ищет в проекте все файлы моделей и отображает из список, сортированный по папкам.
Поддержанные форматы :

1. .prefab
2. .max
3. .fbx

- При клике на модели, она отображается с окне Project View

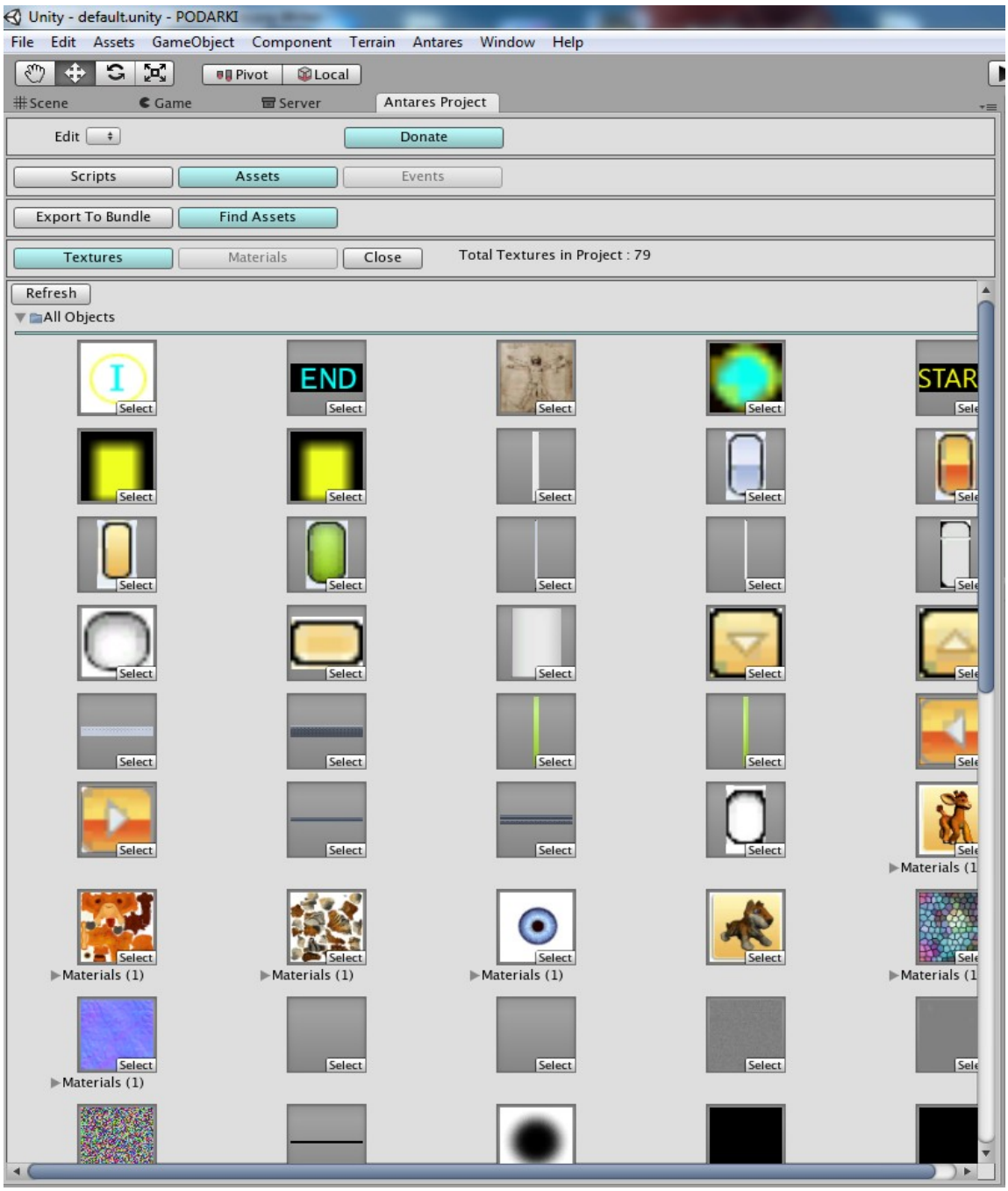
ПРИМЕЧАНИЕ : *Броузер Моделей имеет возможность экспорта в **Asset Bundles**, но более удобный и гибкий экспортёр находиться в* Antares Basket**.**



БРОУЗЕР ТЕКСТУР.

Броузер Текстур отображает список всех Текстур проекта.

- Поддерживаемые форматы : Все форматы **Unity.**
- Клик левой кнопкой мыши : Отображает текстуру в окне Project View.
- Клик правой кнопкой мыши : Открывает Инспектор Текстуры.
- **Drag & Drop** : Вы можете использовать перетаскивание из Броузера в ваши Скрипты и Компоненты.
- Materials : Под каждой текстурой отображён список Материалов, её использующих.
- **Ctrl + Z (Windows)** : Отмена действий по назначению текстур на Материалы, Скрипты или Компоненты работает.



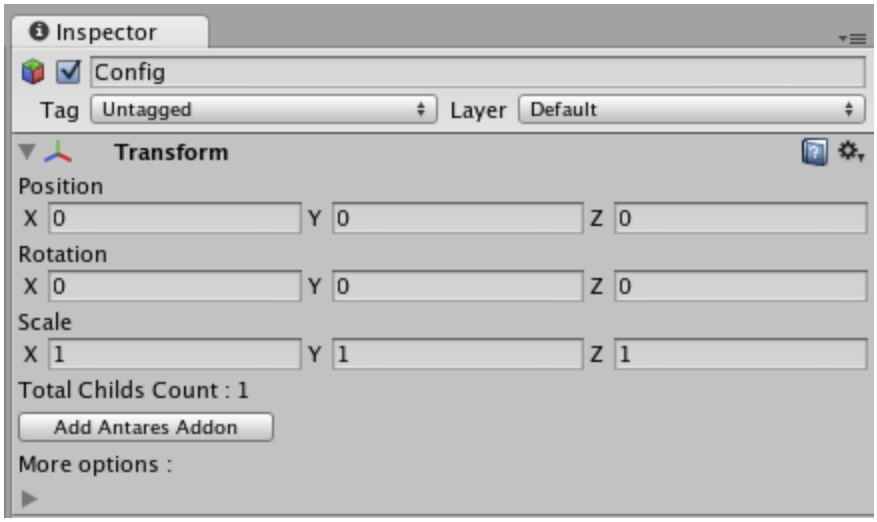
ANTARES ADDON

ANTARES ADDON FEATURES

Antares Addon добавляет новые возможности в стандартный Инспектор компонента Transform :

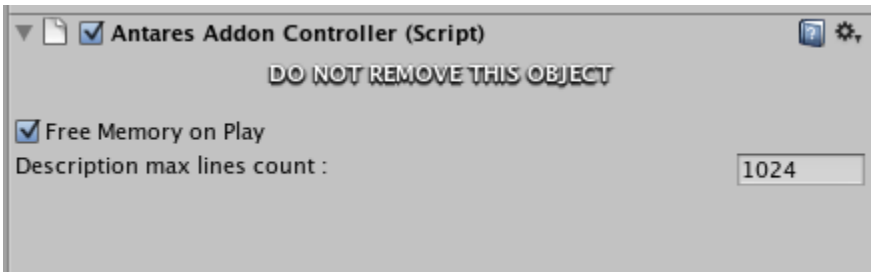
- Total Childs Count : отображает колличество объектов, подчинённых данному объекту.
- Add Antares Addon : Добавляет к объекту компонент AntaresAddon
- More Options : Раскрывающийся список (foldout) содержащий Инструменты для Редактирования Уровней (Level Design)

ANTARES ADDON COMPONENT



- При нажатии кнопки **Add Antares Addon** к объекту автоматически добавляется компонент AntaresAddon.
- Если это первый компонент в вашем проекте, в окне Hierarchy автоматически создаётся объект AntaresController, необходимый для сохранения некоторых параметров ваших объектов и настроек Antares Addon. Объект создаётся только один раз. Не удаляйте его.

НАСТРОЙКИ КОМПОНЕНТА **ANTARES ADDON**



При первом добавлении компонента **AntaresAddon** к какому-либо объекту, в Сцене автоматически создаётся **GameObject**, с прикреплённым к нему компонентом **AntaresAddonController**

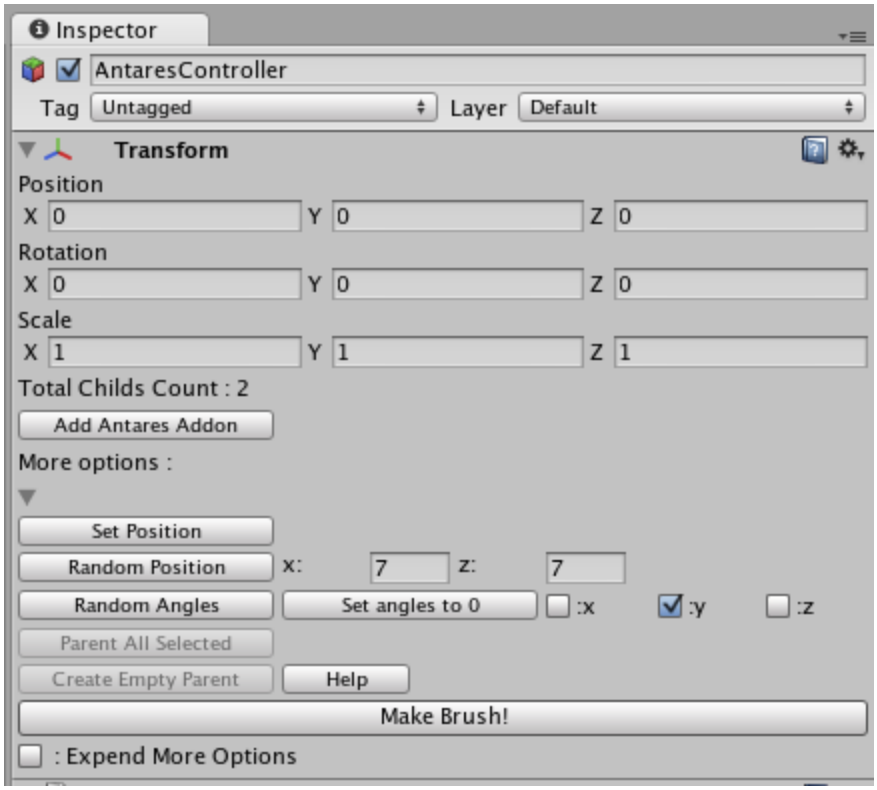
Этот скрипт настраивает работу всех **AntaresAddon** компонентов в Сцене. Он содержит два настраиваемых параметра :

- **Free Memory On Play** (по-умолчанию : включен) : При старте Сцены в режиме Play, очищает переменные всех компонентов Antares Addon, не влияющие на работу программы. Добавлено для экономии памяти.
- **Description max lines count** (по умолчанию : 1024) : Максимальное число символов в поле Object Description компонентов AntaresAddon.

Объект AntaresController содержит подобъект (child) по имени **MultiTagsCollector** Этот объект необходим для хранения информации о мультитагах (multitags) объектов Сцены.

ПРИМЕЧАНИЕ : *Не удаляйте и не изменяйте **AntaresController** и подчинённые ему объекты.*

ИНСТРУМЕНТЫ РЕДАКТИРОВАНИЯ УРОВНЕЙ (LEVEL DESIGN)



- **Set Position** : Устанавливает объект в Сцене по клику мыши. Объекты Сцены должны иметь коллайдеры (Collider)
- **Random Position** : Случайно смешает объект в сцене в пределах заданного диапозона.
- **Random Angles** : Случайное вращение объекта.
- **Set angles to 0 :** устанавливает градус объект в 0 по выбранной оси.
- **Parent All Selected :** Создаёт пустой GameObject с именем _parent и приаттачивает высе выбранные объекты к нему.
- **Create Empty Parent :** Создаёт пустой GameObject, устанавливает его в центр меша выбранного объекта и приаттачивает объект к новому GameObject

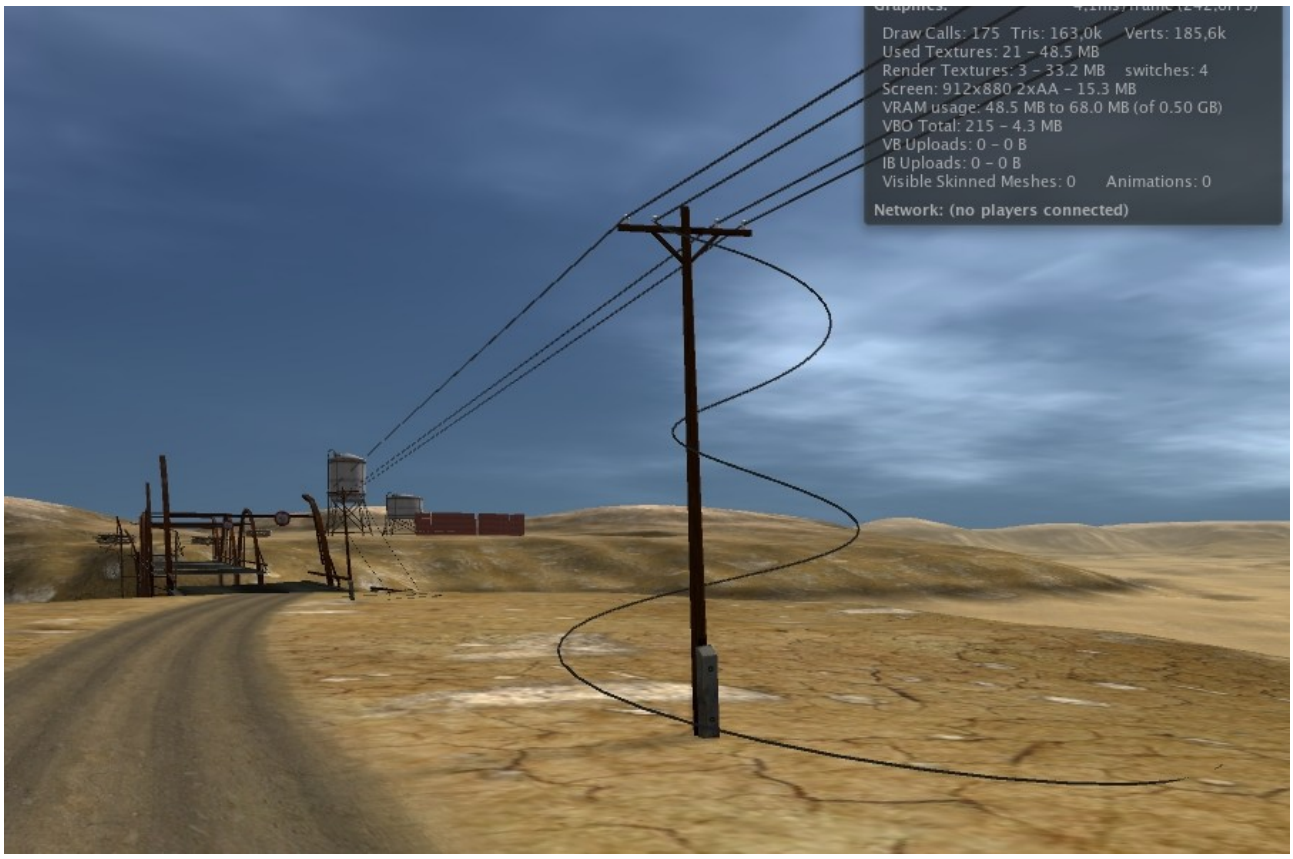
ПРИМЕЧАНИЕ : *Работает только с объектами имеющими Mesh. Удобно для тех случаев, когда Pivot объекта находится далеко от его визуального центра.*

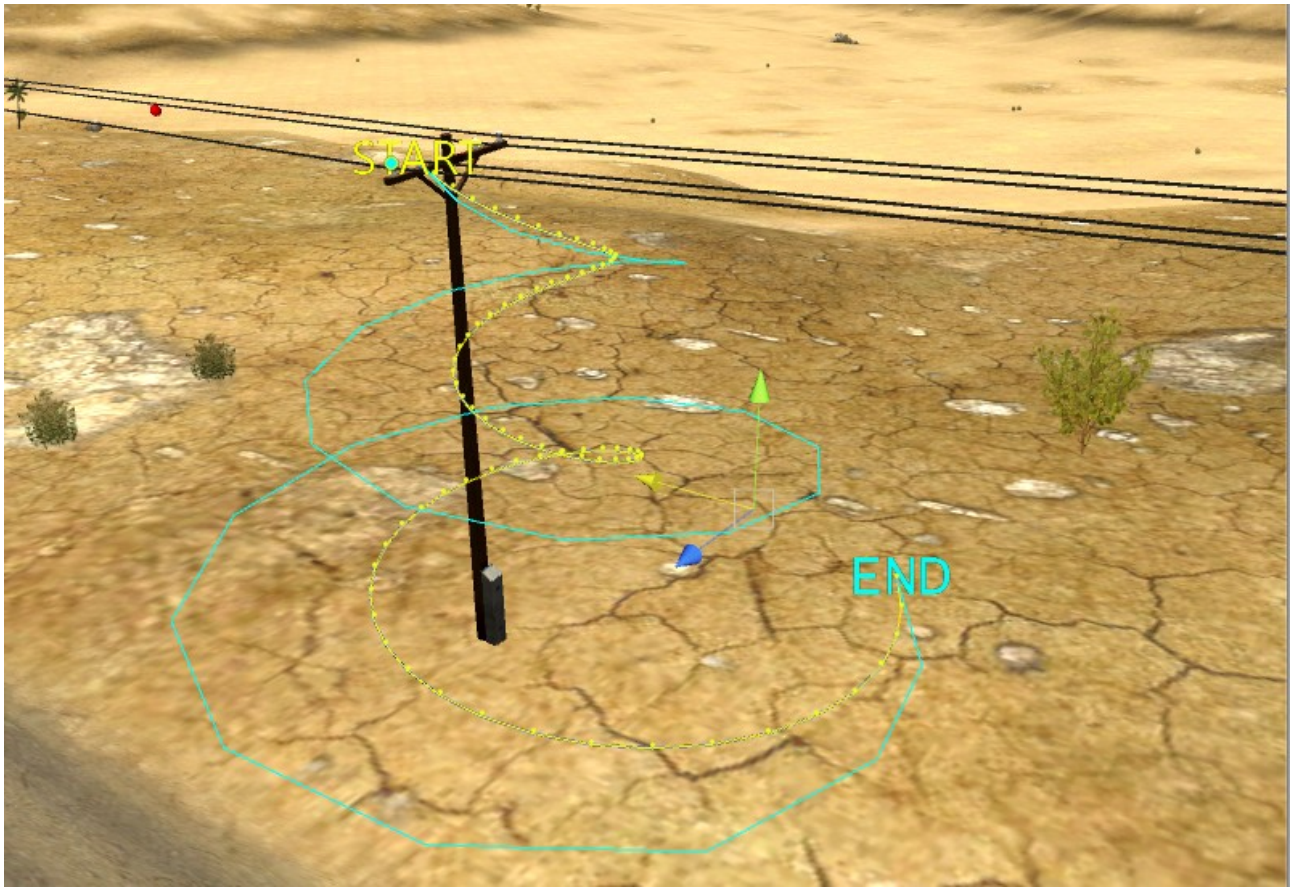
- **Make Brush! :** Создаёт кисть из выбранных объектов и позволяет по клику мыши создавать неограниченное колличество копий объектов в сцене.
- **Expend More Options :** Если включено, раскрывающийся список (foldout) **More options** будет открыт во всех Инспекторах всех объектов в Проекте.

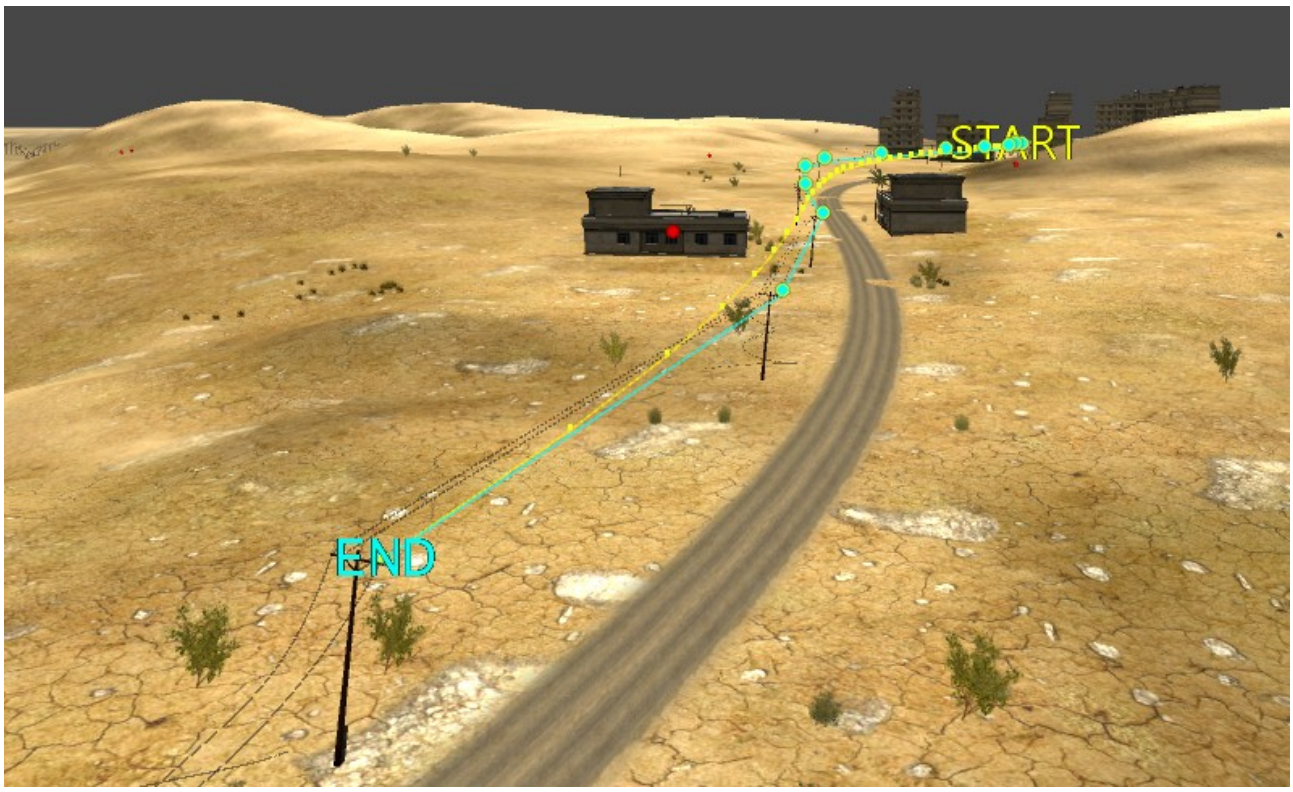
ПРИМЕЧАНИЕ : *Чтобы отключить Кисть, повторно нажмите на кнопку* ***Make Brush!***

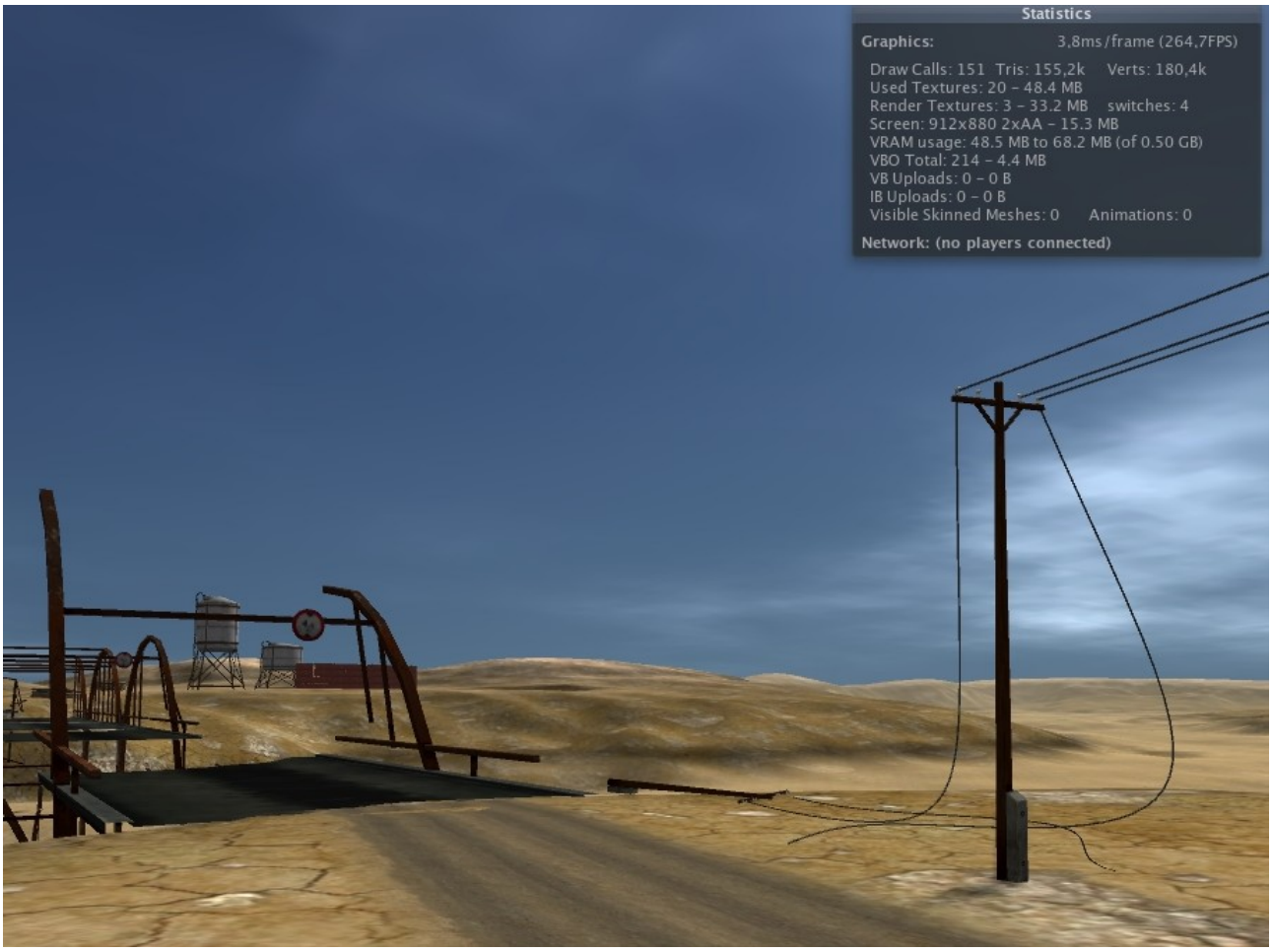
КРИВЫЕ БЕЗЬЕ (BEZIER CURVE)

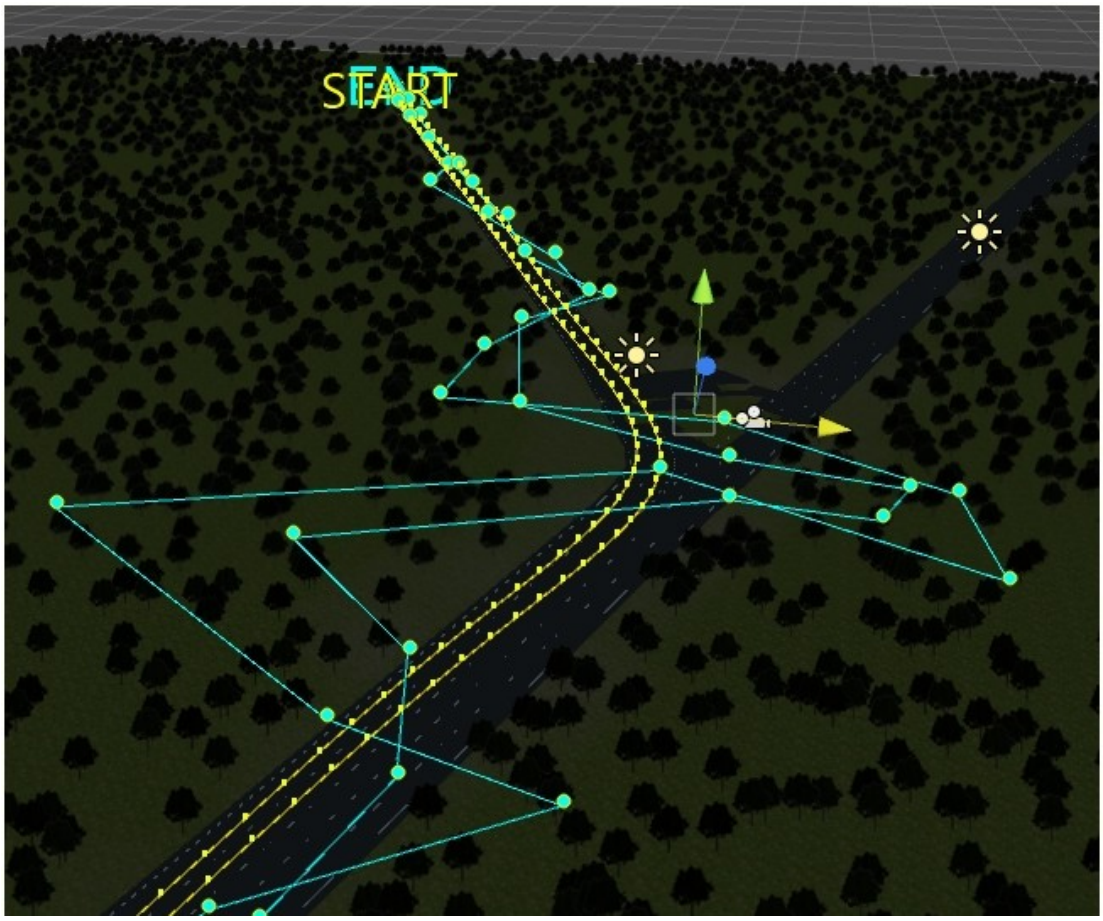
Кривые Безье очень важный и удобный инструмент Antares Addon. С их помощью вы можете прокладывать трассы движения автомобилей, или NPC, сами дороги (Antares Curved Road) или создавать реалистичные электросети в своей игре. В виду того, что для создания проводов можно использовать компонент Line Renderer это самый дешёвый и продуктивный путь для придания неповторимого разнообразия вашей игре.


Draw Calls: 175 Tris: 163,0k Verts: 185,6k
Used Textures: 21 – 48.5 MB
Render Textures: 3 – 33.2 MB switches: 4
Screen: 912x880 2xAA – 15.3 MB
VRAM usage: 48.5 MB to 68.0 MB (of 0.50 GB)
VBO Total: 215 – 4.3 MB
VB Uploads: 0 – 0 B
IB Uploads: 0 – 0 B
Visible Skinned Meshes: 0 Animations: 0
Network: (no players connected)


START
END


START
END


Statistics
Graphics: 3,8ms/frame (264,7FPS)
Draw Calls: 151 Tris: 155,2k Verts: 180,4k
Used Textures: 20 - 48.4 MB
Render Textures: 3 - 33.2 MB switches: 4
Screen: 912x880 2xAA - 15.3 MB
VRAM usage: 48.5 MB to 68.2 MB (of 0.50 GB)
VBO Total: 214 - 4.4 MB
VB Uploads: 0 - 0 B
IB Uploads: 0 - 0 B
Visible Skinned Meshes: 0 Animations: 0
Network: (no players connected)

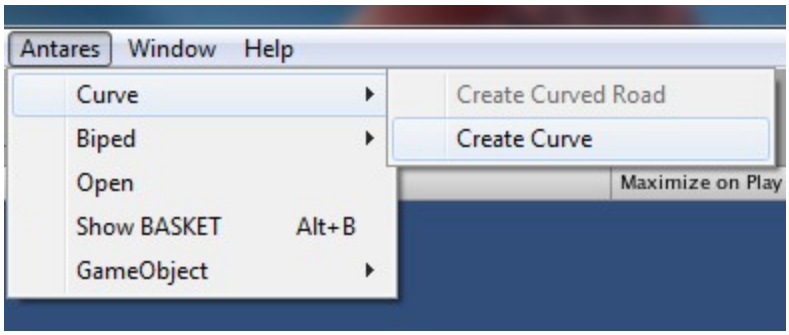

START


Начало инцидента
8 : 34 : 0

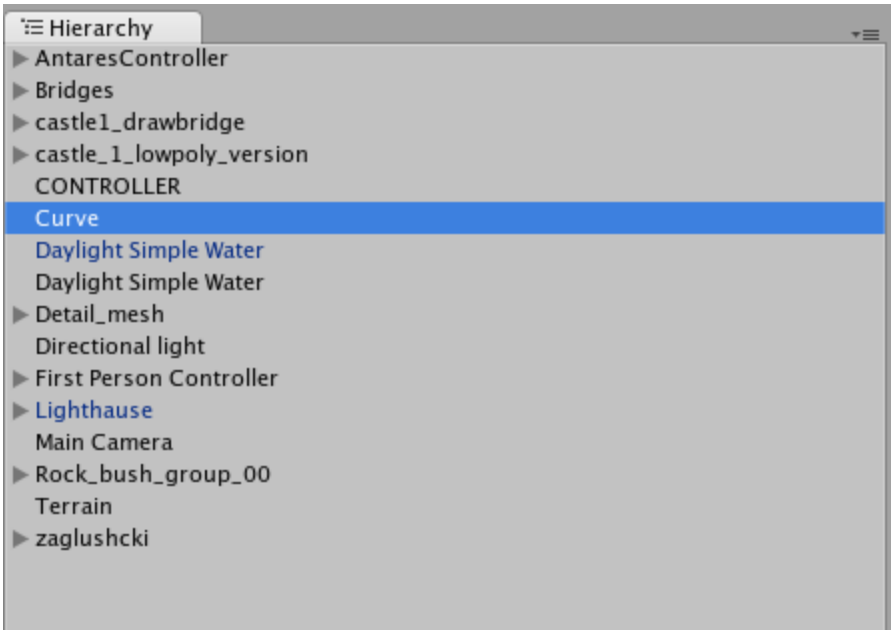
Итак, научимся создавать Кривые в Редакторе.

СОЗДАНИЕ КРИВЫХ В РЕДАКТОРЕ UNITY

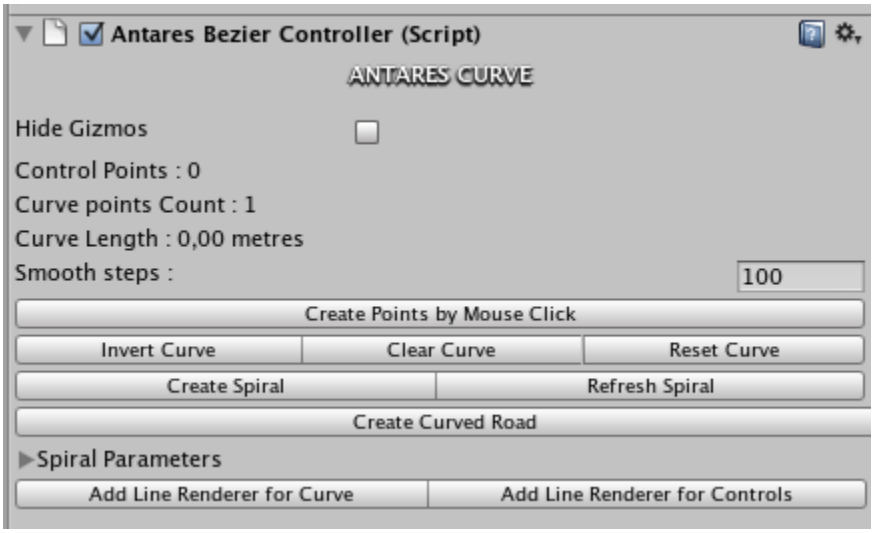
Чтобы содать Кривую, выбираем **Antares → Curve → Create Curve**



В окне **Hierarchy** появится новый объект **Curve**.



В Инспекторе этого объекта мы увидим :



- **Hide Gizmos** : Включает и выключает отображение контрольных точек Кривой.
- **Control Points** : колличество Контрольных Точек Кривой
- **Curve Points** : Колличество точек в Кривой
- **Curve Length** : длина Кривой в метрах.
- **Smooth Steps** : Максимальное колличество точек в Кривой.
- **Create Points by Mouse Click :** если эта кнопка активна (подсвечена голубым цветом), вы можете строить свою Кривую, устанавливая Контрольные Точки в местах кликов в окне Scene.

Чтобы отключить этот режим, снова нажмите на кнопку Create Points by Mouse Click.

- **Invert Curve :** Поменять местами Начало (Start) и Конец (End) Кривой.
- **Clear Curve :** Удалить Контрольные Точки и Точки Кривой.
- **Reset Curve (Только для Spiral) :** сбрасывает настройки Спирали на исходные.
- **Create Spiral :** Создаёт Спираль.
- **Refresh Spiral :** перерасчёт Спирали, после изменений её настроек.
- **Create Curved Road :** Добавляет к Кривой компонент CurvedRoad, необходимый для генерации дорог.

- **Раскрывающийся список (foldout) Spiral Parameters :** Настройки Спирали. Будут рассмотрены ниже.
- **Add Line Renderer for Curve :** Добавляет компонент Line Renderer, создающий Line Renderer по Точкам Кривой
- **Add Line Renderer for Controls :** Добавляет компонент Line Renderer, создающий Line Renderer по Контрольным Точкам.

РЕДАКТИРОВАНИЕ КРИВОЙ В РЕДАКТОРЕ UNITY Давайте создадим Кривую на одной из дорогой этой Сцены :

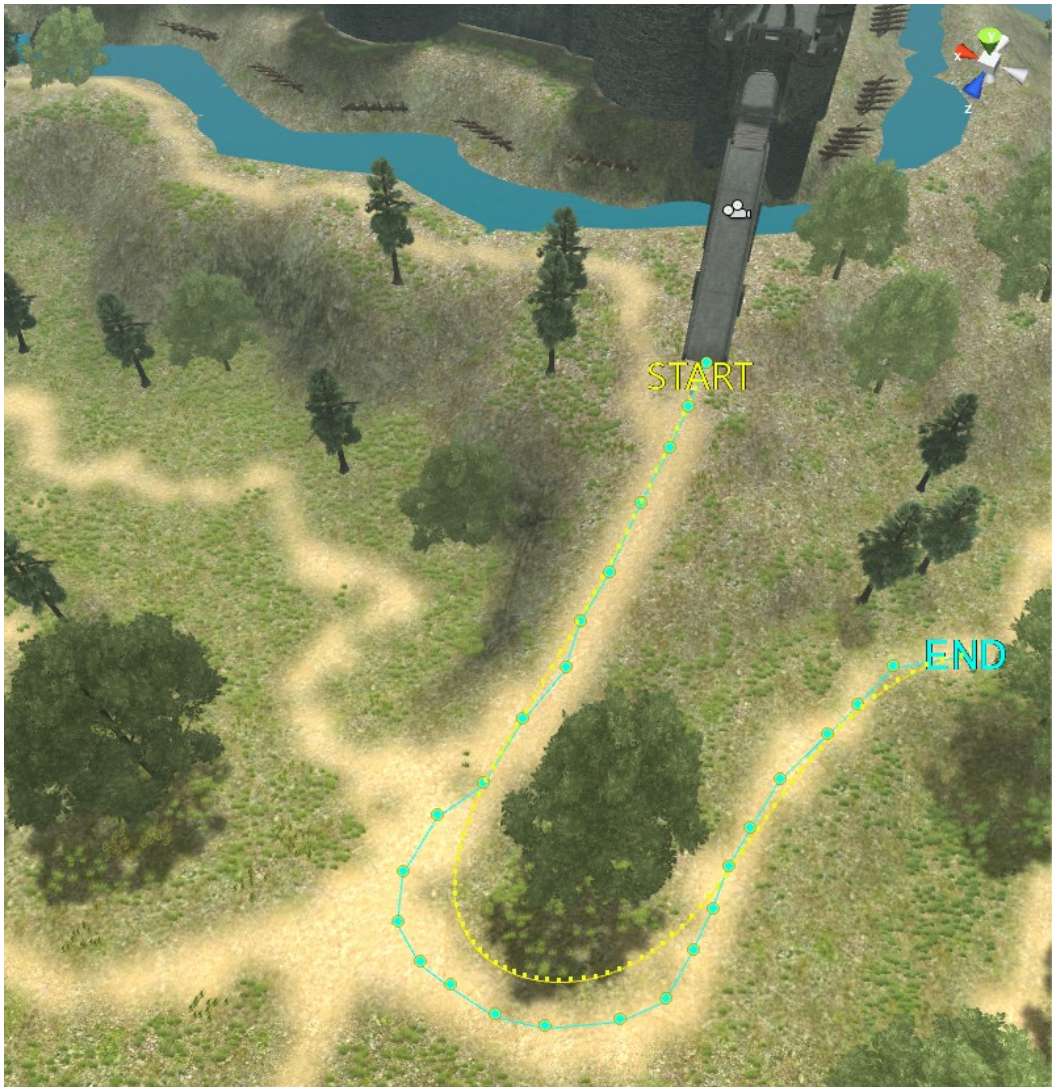
- Нажимаем кнопку Create Points by Mouse Click.
- Кнопка подсвечивается голубым цветом.
- Теперь мы готовы проложить контрольные точки нашей Кривой в Сцене.

ПРИМЕЧАНИЕ : *В местах, где Кривая делает поворот, добавляйте больше Контрольных Точек.*

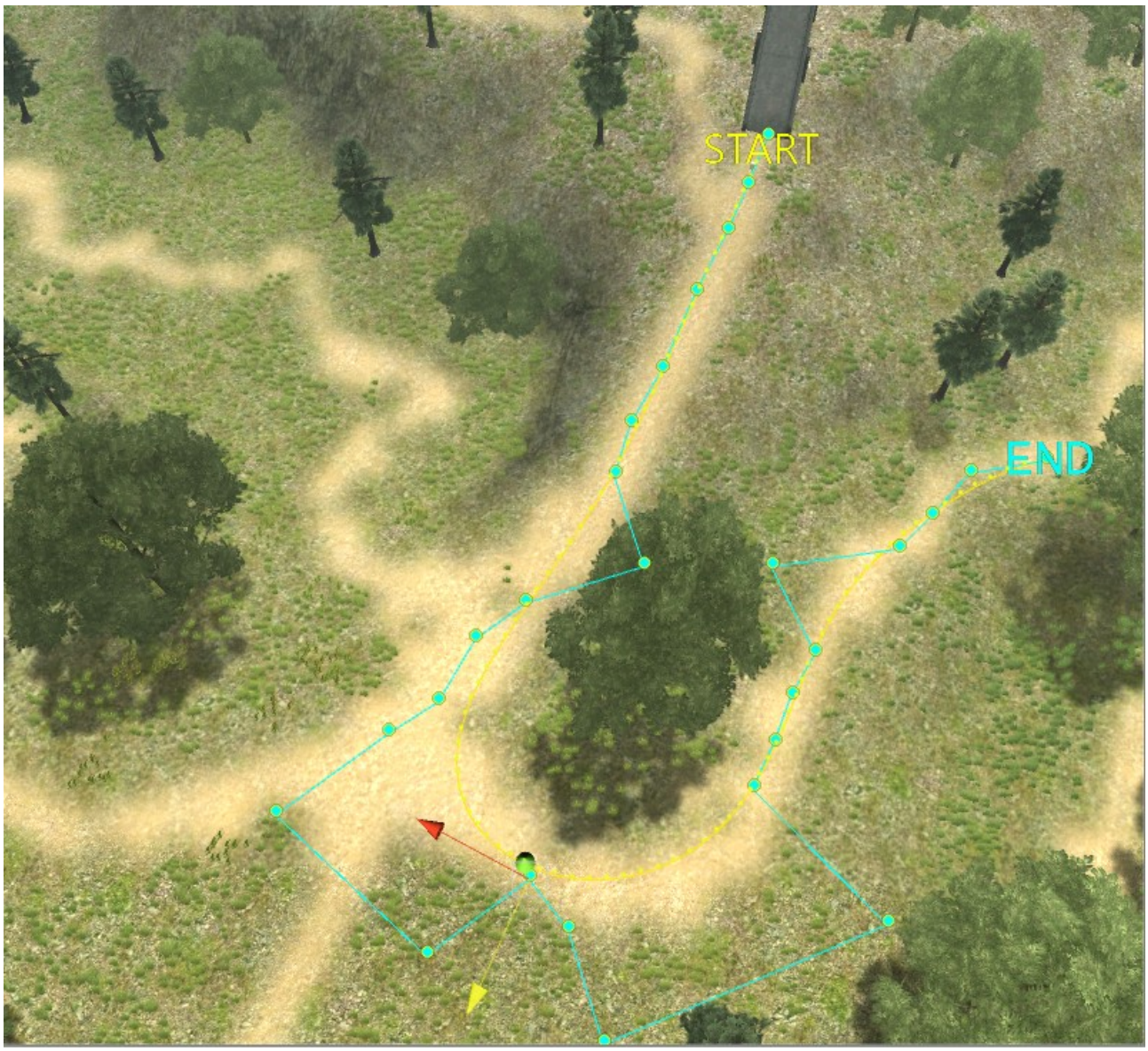
ПРИМЕЧАНИЕ : *Иногда Кривая может потерять фокус и вместо её Инспектора, вы увидите Инспектор последней поставленной Контрольной Точки. Просто кликните на кнопке* ***Go To The Root*** *и вы вернётесь в Инспектор Кривой и сможете продолжить расставление Контрольных Точек.*

- Голубые точки, связанные голубыми линиями это Контрольные Точки, контроллирующие изгибы Кривой.
- Жёлтые точки с жёлтыми линиями это Точки Кривой или Точки Сглаживания, отображающие саму Кривую.

Закончив расставление Контрольных Точек, мы получим примерно следующее :



Чтобы отредактировать Кривую, мы выделяем нужную контрольную точку и перемещаем её как обычный GameObject в окне Scene



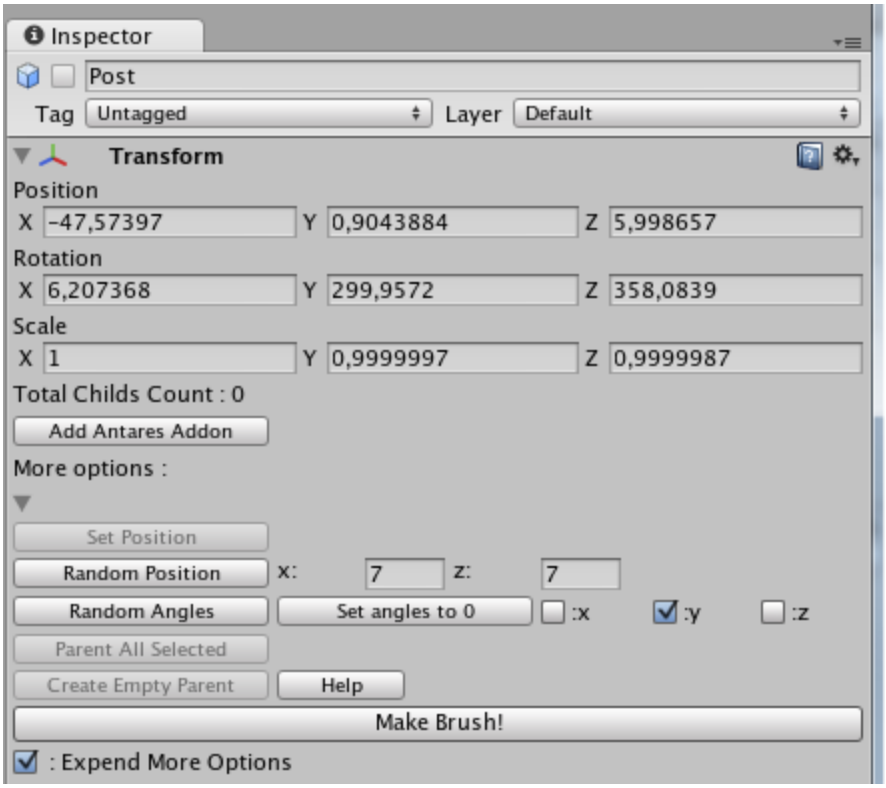
Поздравляю! Наша Кривая построена! Теперь мы можем создать из неё дорогу.

СОЗДАНИЕ ПРОВОДОВ

Теперь рассмотрим простой и быстрый способ создания электрических проводов. Предположим, что мы имеем несколько электроопор и хотим соединить их проводами.

- Добавим к опорам компонент MeshCollider (нам понадобятся объекты, в которые мы сможем ткнуть мышью).
- Создадим Кривую. **Antares → Curve → Create Curve.**
- Расставим в сцене столбы (опоры)
  - Добавим в Сцену один столб
  - Нажмем в Инспекторе кнопку More options → Expend More Options

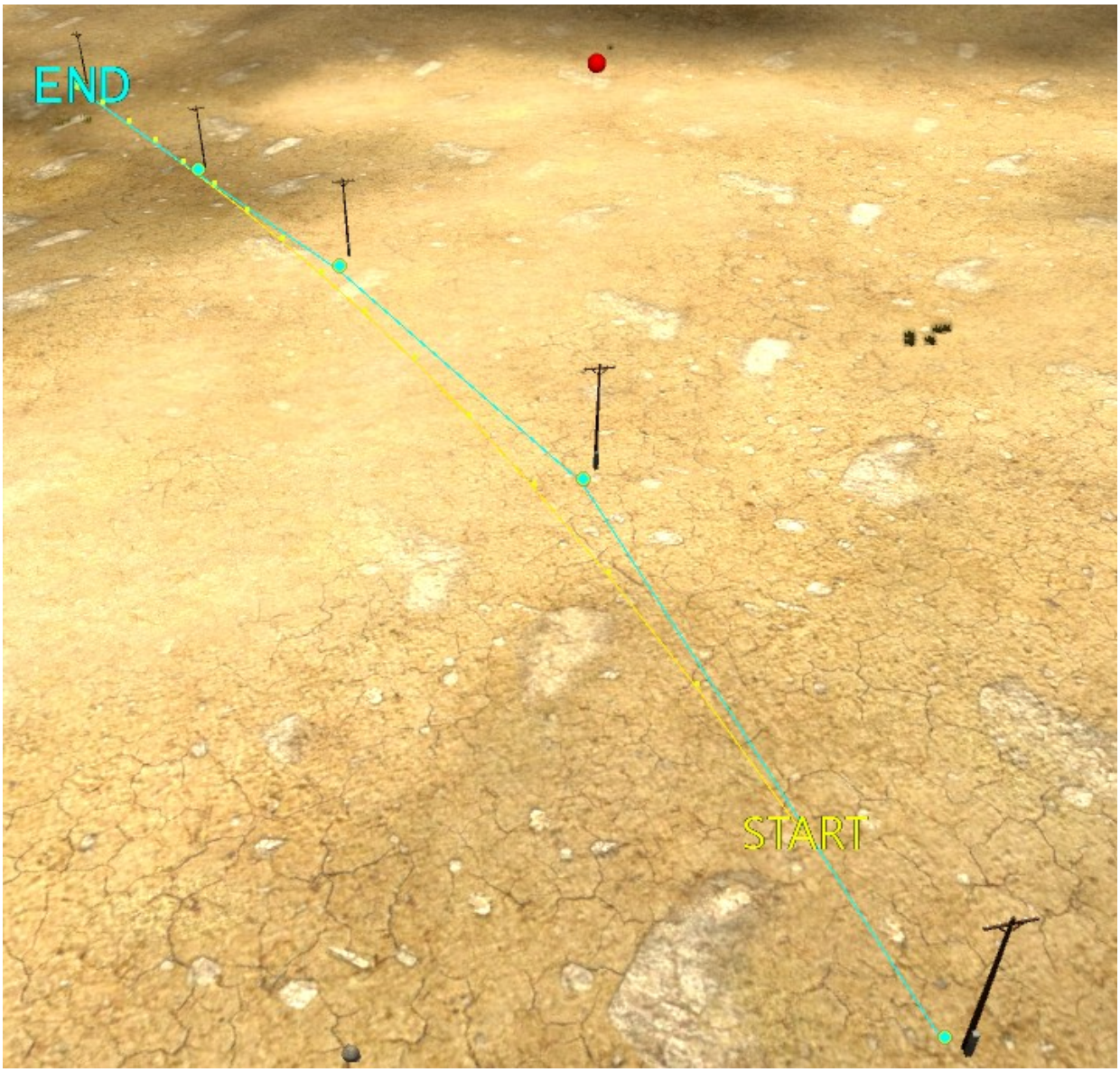
ПРИМЕЧАНИЕ : *Это нужно для того, чтобы раскрывающийся список (foldout) был открыт постоянно на всех объектах.*



- Включим кнопку **Make Brush!**
- Кликая в Scene, расставим цепочку столбов.
- Отключим кнопку **Make Brush!**

- Теперь пришла очередь Кривых.
  - Выберем нашу Кривую.

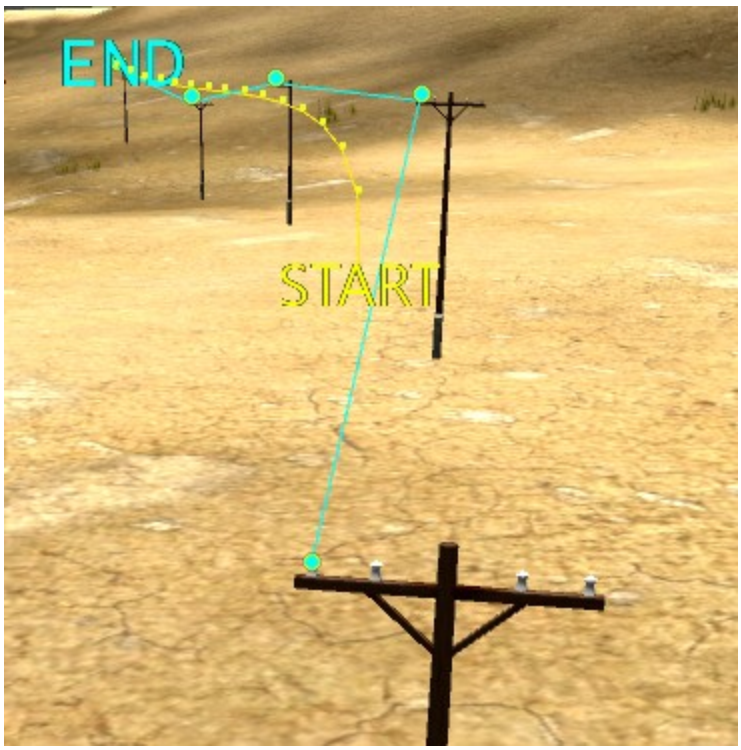
- Включим режим **Create Points by Mouse Click**
- Расставим 5 контрольных точек возле оснований столбов.



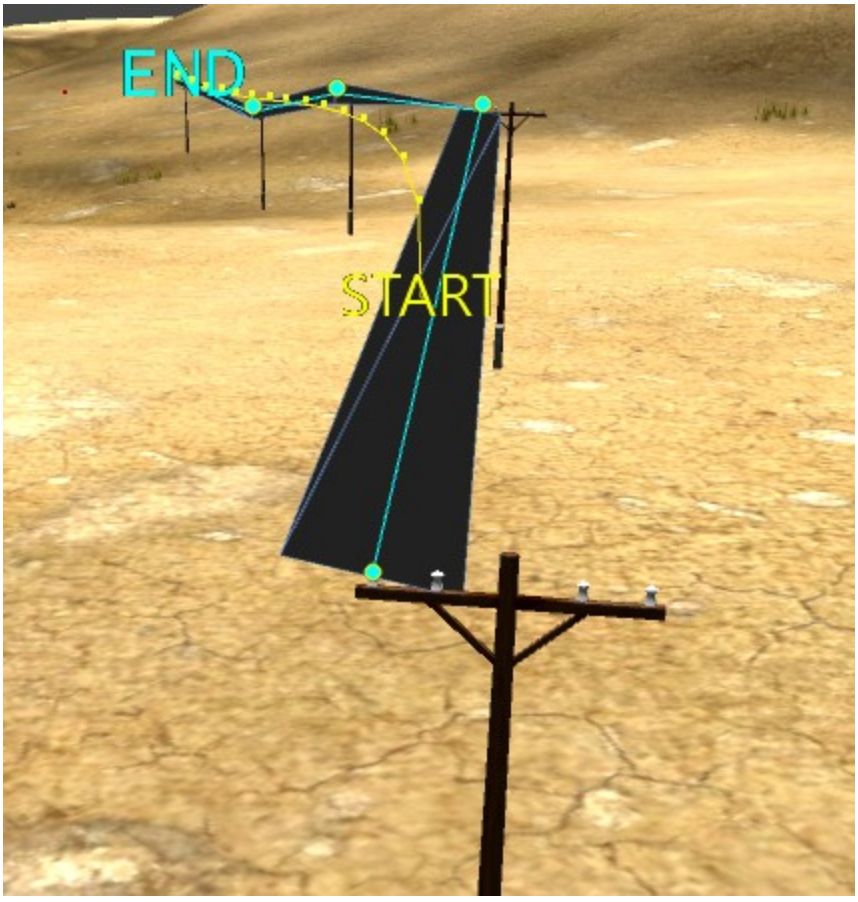
- Отключим режим **Create Points by Mouse Click**
- Выберем ближайщую Контрольную Точку (голубой кружок) и поставим камеру Сцены так, чтобы удобно видеть столб.

- Нажмём кнопку **Set Position**.
- Кликнем туда, где мы хотим установить наш провод.

- Последовательно повторим эти действия с остальными Точками и столбами.
- Через три минуты, всё готово и все пять столбов соединены Контрольными Точками Кривой.

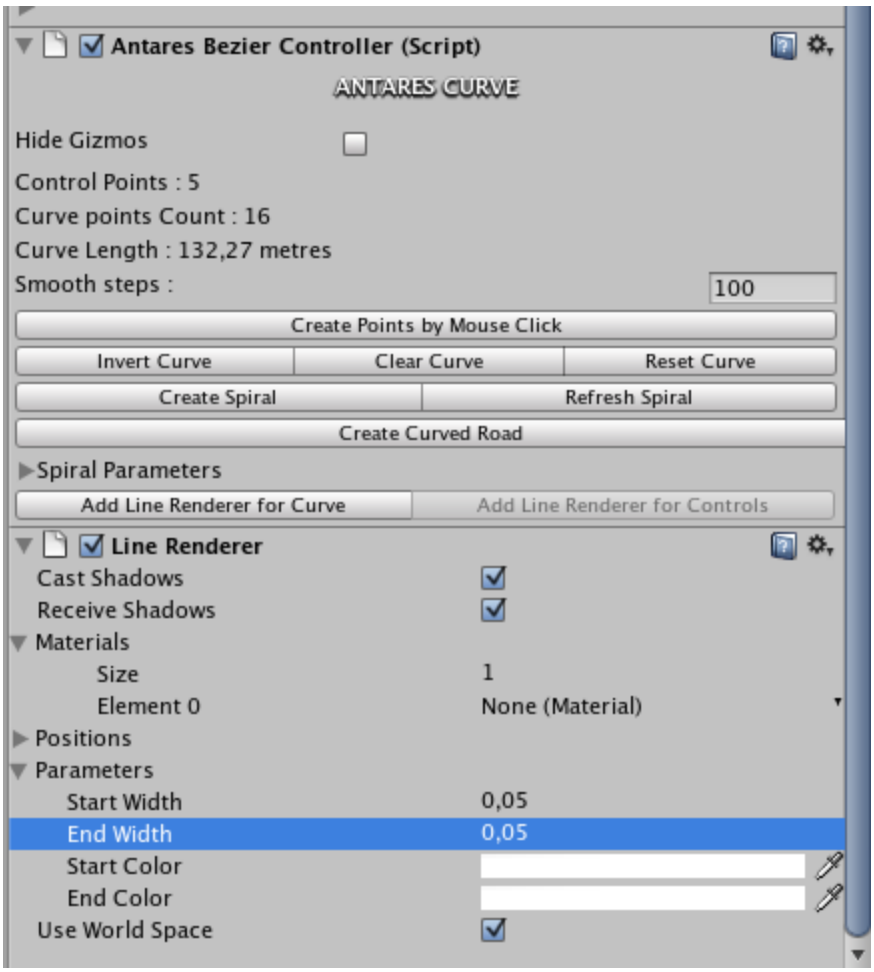


- Добавим к Кривой компонент Line Renderer.
  - Нажмём кнопку **Add Line Renderer for Controls**

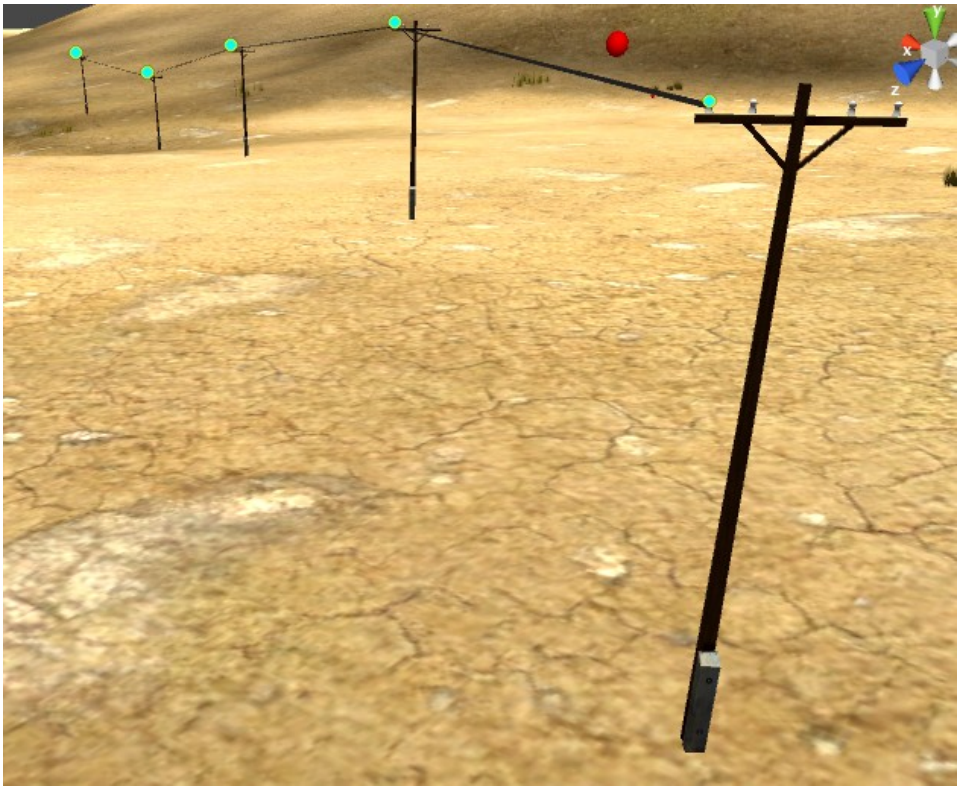


Такая толстая линия мало похожа на провод.

- Настроим её толщину, установив примерно такие параметры :



Поздравляю! Первый провод готов!



Далее, просто сделайте дубли Кривой (Ctrl + D (Windows)) и установите их на нужные места.

Создание линии электропередач закончено.

СПИРАЛИ

Создать спираль так же просто, как Кривую, только ещё проще.

- Добавим Кривую в Сцену : Antares → Curve → Create Curve
- Выберем в Инспекторе кнопку Set Position и установим Кривую в то место, где мы хотим её видеть.
- Нажмём в Инспекторе кнопку Create Curve

Спираль построена.

Если добавить к ней компонент Line Renderer (**Add Line Renderer for Controls)**, то мы сможем получить вот такой изумительный оборванный провод.

НАСТРОЙКИ СПИРАЛЕЙ (SPIRAL SETTINGS)

- Spiral Height : Высота Спирали
- Bottom Radius : Радиус нижнего витка Спирали.
- Up Radius : Радиус верхнего витка Спирали.
- Rounds : Колличество витков Спирали.
- Control Points Count : Колличество Контрольных Точек Кривой Спирали.
- Clock Wise Direction : Если включено, спираль завивается по часовой стрелке.

ПРИМЕЧАНИЕ : *При изменении настроек, обязательно нажмите кнопку* ***Refresh Spiral***

ДИНАМИЧЕСКОЕ СОЗДАНИЕ И РЕДАКТИРОВАНИЕ КРИВЫХ.

Кривые можно создавать или изменять прямо в runtime режиме. Для этого существует интерфесный класс Curve (Curve.cs)

CURVE API

Скачать Демо-проект
Посмотреть ONLINE

Чтобы создать Кривую в режиме runtime, используйте следующий код :

```
GameObject myCurve = new GameObject("My Dinamic Curve", typeof(AntaresBezierController));
```

КОНСТРУКТОР И ДЕСТРУКТОР

public Curve(GameObject curveObject) : Конструктор. В качестве аргумента принимает GameObject, содержащий Кривую. На пример : myCurve или объект, созданный ранее в Редакторе **Unity**.

public void Dispose() : Деструктор. Вызовите его, если хотите уничтожить интерфесный объект Curve (Кривая **не будет** уничтожена автоматически).

ПЕРЕМЕННЫЕ :

public float curveLength : Длина Кривой в метрах.
public Vector3[] curvePoints : массив векторов Точек Кривой (Точек Сглаживания).

КОНТРОЛЬНЫЕ ТОЧКИ

public int GetCurveControlPointsCount() : Возвращает колличество Контрольных Точек Кривой.

public void AddCurveControlPoint(Vector3 postion, bool refreshCarve) : Добавляет новую Контрольную Точку.
Vector3 postion : позиция Контрольной Точки в глобальных координатах.
bool refreshCarve : Если true, Кривая будет обновлена и перерасчитана.

public void RemoveCurveControlPoint(int indexOfPoint, bool refreshCarve) : Удаляет Контрольную Точку.
int indexOfPoint : номер Контрольной Точки.
bool refreshCarve : Если true, Кривая будет обновлена и перерасчитана.

public GameObject GetNearestControlPoint(Vector3 position) : Возвращает Контрольную Точку, ближайщую к Vector3 position

ТОЧКИ КРИВОЙ (Точки Сглаживания)

public int GetCurveSmoothPointsCount() : Возвращает колличество точек Кривой (Точек Сглаживания).

public int GetNearestCurvePoint(Vector3 position) : Возвращает ближайщую точку к Vector3 position

public int GetNearestCurveSmoothPoint(float position) : Возвражает индекс ближайщей Точки Кривой (Точки Сглаживания) в массиве Точек Кривой.
float position : Позиция на Кривой в диапазоне от 0.0 до 1.0

public void RefreshCurve() : Обновить и перерасчитать Кривую.

public void RefreshLineRenderers() : Обновить и перерасчитать параметры присоединённых компонентов Line Renderer

public void InvertCurve() : Поменять порядок Контрольных Точек. Меняет местами Начало (Start) и Конец (End) Кривой.

public float GetCurveLength(bool calculateAgain) : Возвращает длину Кривой в метрах.

KNOWN BUGS (ИЗВЕСТНЫЕ ОШИБКИ)

1. К сожалению, я плохой математик, поэтому для генерации Кривых использовал алгоритмы, которые нашёл в интернете. Не самые лучшие. Главный минус этого алгоритма в том, что если колличество Точек Кривой (Точек Сглаживания) превышает 30, далее Кривая строится некорректно. Поэтому я ввёл алгоритм последовательного расчёта Кривых Безье (Bezier Splines), группами по 30 точек. Из за этого, если очередной «разрыв расчёта» приходится на сгиб Кривой, в этой точке возможно появление острого угла. Вы можете сгладить его с помощью Контрольных Точек.

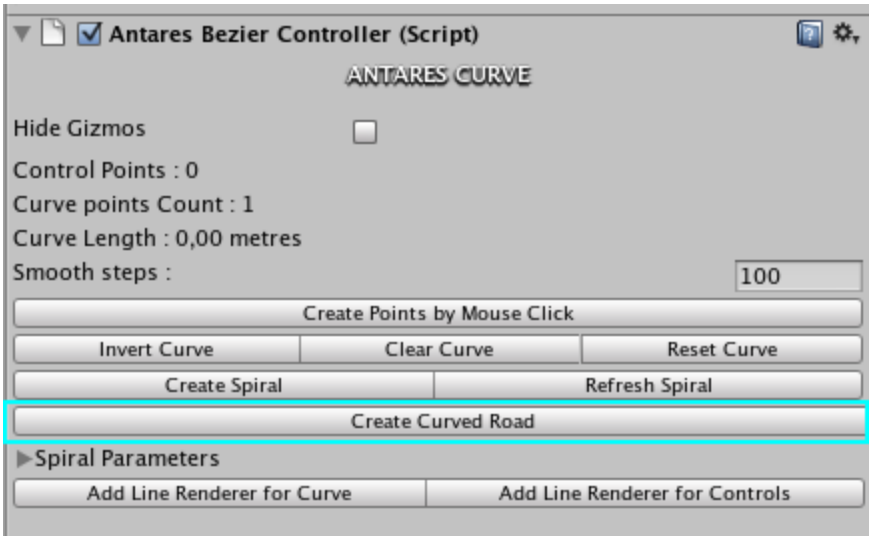
2. Иногда Кривая может потерять фокус и вместо её Инспектора, вы увидите Инспектор последней поставленной Контрольной Точки. Просто кликните на кнопке **Go To The Root** и вы вернётесь в Инспектор Кривой и сможете продолжить расставление

Контрольных Точек.

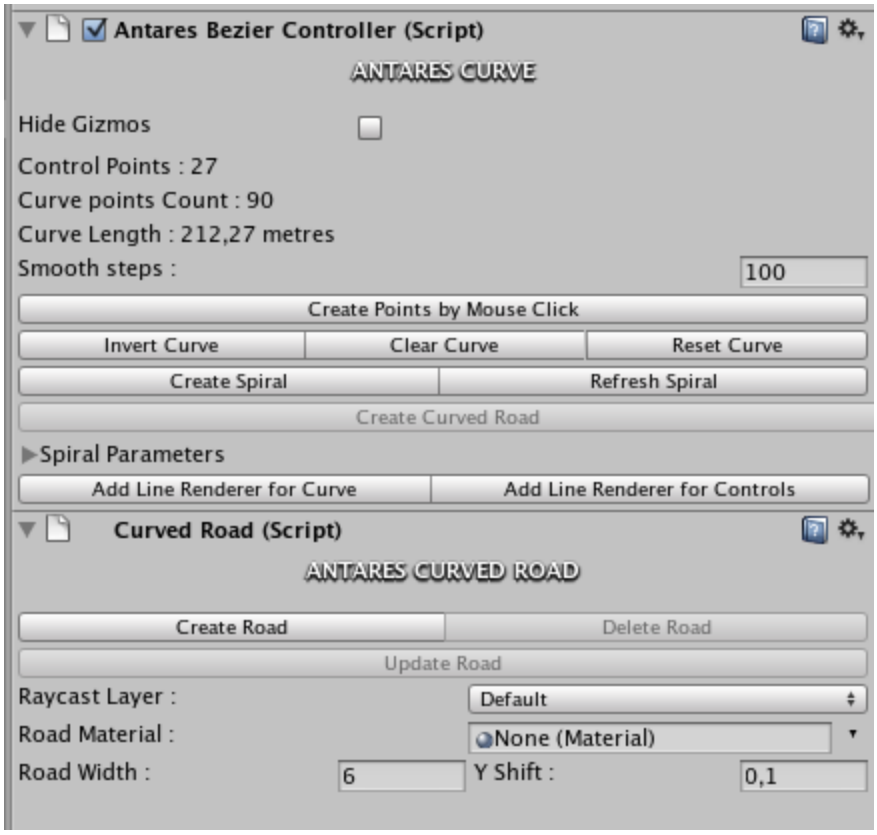
ГЕНЕРАТОР ДОРОГ НА БАЗЕ КРИВЫХ.

При помощи **Antares Curve** вы можете проложить дороги в вашей игре. Чуть выше мы рассмотрели пример создания Кривых. Теперь продолжим его и создадим Дорогу.

Итак, мы проложили Кривую по тропинке. Теперь нажмём кнопку Create Curved Road



К нашей Кривой добавился компонент **Curved Road**



- **Raycast Layer :** Слой (layer) в котором будет производится поиск коллизий для ровной и корректной расстановки вершин генерируемого меша дороги.
- **Road Material :** Материал дороги.
- **Road Width :** Ширина дороги в метрах.
- **Y Shift :** Небольшое смещение дороги вверх, для того, чтобы она не оказалась внутри террейна (terrain) или объекта, на котором она создаётся.
- **Update Road** : Нажать для применения настроек к уже построенной дороге.

После настройки параметров, нажимаем кнопку Create Road.
Поздравляю! Ваша первая дорога создана!

1. **ПРИМЕЧАНИЕ :** *На созданной дороге мы видем несколько дыр. Это не дыры. Это так называемый эффект Z-fighting. При виде сверху, рендерер* ***Unity*** *не всегда корректно может расчитать положение вершин в пространстве и прорисовывает Terrain выше дороги. При виде снизу этот эффект пропадает. Вы можете попробовать приподнять дорогу чуть выше (настройка Y Shift), но не забудьте нажать кнопку* **Update Road**.

2. **ПРИМЕЧАНИЕ :** *Иногда сквозь полотно дороги, действительно может проглядывать часть террейна (Terrain). Просто сгладьте это место кистью, чтобы убрать резкие возвышенности и нажмите кнопку* ***Update Road.*** *Так же вы можете увеличить параметр Smooth Steps Кривой. Это увеличит число Точек Сглаживания и, следовательно, увеличит колличество полигонов дороги, что вполне может скрыть неровности.*

ГЕНЕРАТОР ДОРОГ : ШЕЙДЕРЫ

Только что созданная нами дорога, выглядит как асфальтовое шоссе. А в данной сцене, больше пригодилась бы тропинка. Что же делать? Конечно же, использовать шейдеры.

Что нам нужно? Размыть край дороги и добавить к ней текстуру.

Берём текстуру с альфа-каналом по краям. К примеру, вот такую :

К сожалению, шейдеры писать я не очень умею, поэтому моих знаний хватило только на такой вариант :

Вы можете скачать шейдер дороги тут.

После манипуляции с материалом и шейдером, наша дарога приобретает такой вид :

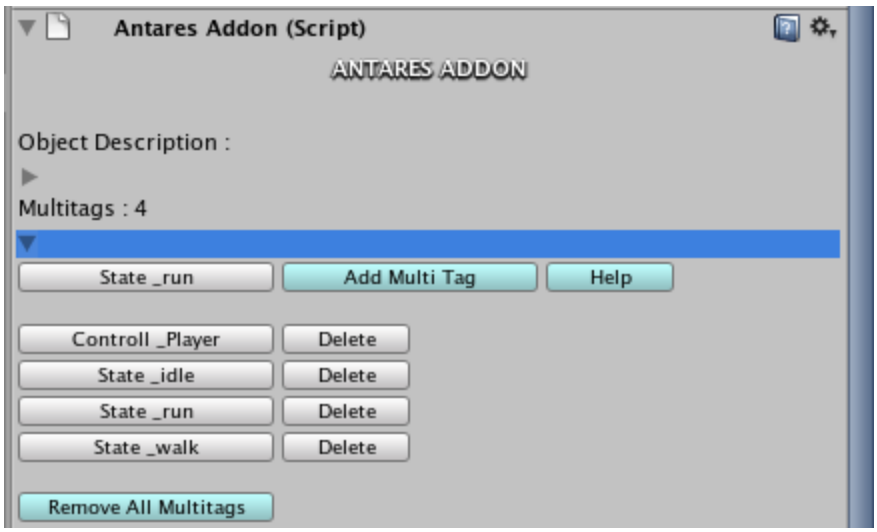
ПРИМЕЧАНИЕ : *Исправленный мною шейдер, не принимает тени. И, к сожалению, наша дорога тоже. Надеюсь, кто-то из вас, дорогие друзья, напишет новые, лучшие варианты шейдера для дорог.*

СИСТЕМА МУЛЬТИТАГОВ (MULTITAGS)

Ещё одна возможность Antares Addon : Система мультитагов. Благодаря ей, любой объект в Сцене может иметь любое колличество Тагов (Tags). Да-да. Именно Тагов **Unity**.

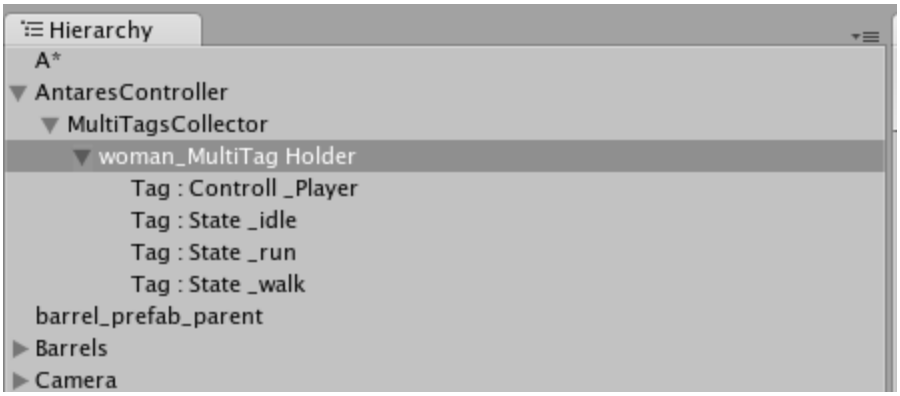
Вы можете использовать для простого и удобного добавления к объекту разнообразных свойств. К примеру один объект может иметь следующие таги :

1. Distructable (Разрушаемый)
2. Cannot float in water (Не может плавать в воде)
3.



ОПИСАНИЕ ПРИНЦИПОВ РАБОТЫ

Когда вы добавляете мультитаг к выбранному объекту, к объекту **AntaresController → MultiTagsCollector** добавляется child с именем **имяобъекта + _MultiTag Holder**



Эти, так называемые *Объекты-Носители Мультитагов* получают выбранный таг и служат для сохранения информации в runtime режиме.
ПРИМЕЧАНИЕ : *Не удаляйте и не изменяйте ничего в структуре объекта AntaresController или в его дочерних объектах.*

- Мультитаги так же могут быть сохранены в префаб (prefab) и автоматически добавлены к объекту при его добавлении в Сцену.
- Мультитаги могут быть добавлены или удалены в режиме runtime через **Multitags API**

MULTITAGS API

Доступ к мультитагам осуществляется через статические методы класса
AntaresAddonController

ПРИМЕЧАНИЕ : *Не используйте **AntaresAddon.cs** для доступа к методам мультитагов. Методы этого класса служебные.*

СЛУЖЕБНЫЕ МЕТОДЫ
(не используйте их.)

public static AntaresAddonController InitController()
public static Transform GetMultiTagsCollector()
public static bool FreeMemoryOnPlay()
public static int GetDescCount()

МЕТОДЫ

public static void AddMultitagToObject(GameObject obj, params string[] mtags)
public static void AddMultitagToObject(Transform obj, params string[] mtags)
Добавить мультитаги к объекту.

public static void RemoveMultitagFromObject(GameObject obj, params string[] mtags)
public static void RemoveMultitagFromObject(Transform obj, params string[] mtags)
Удалить мультитаги с объекта.

public static void RemoveAllMultitagsFromObject(GameObject obj)
public static void RemoveAllMultitagsFromObject(Transform obj)
Удалить все мультитаги с объекта

public static bool ObjectHasMultitag(GameObject obj, string multitag)
public static bool ObjectHasMultitag(Transform obj, string multitag)
Имеет ли объект заданный мультитаг?

public static bool ObjectHasMultitagsBoth(GameObject obj, params string[] multitags)
public static bool ObjectHasMultitagsBoth(Transform obj, params string[] multitags)
Имеет ли объект все проверяемые мультитаги?

public static bool ObjectHasMultitagsAny(GameObject obj, params string[] multitags)
public static bool ObjectHasMultitagsAny(Transform obj, params string[] multitags)
Имеет ли объект любой из заданных мультитагов?

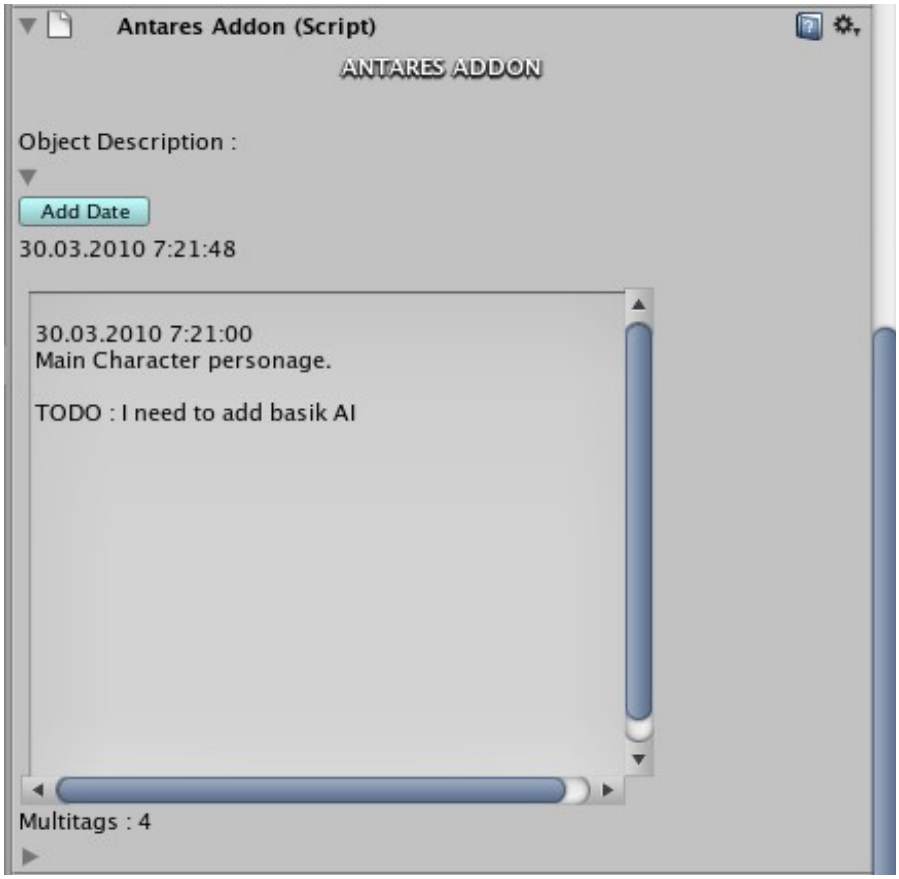
public static int ObjectsMultitagsCount(GameObject obj)
public static int ObjectsMultitagsCount(Transform obj)
Количество мультитагов объекта

public static GameObject GetMultitagObject(GameObject obj, string multitag)
public static GameObject GetMultitagObject(Transform obj, string multitag)
Возвращает объект, хранящий заданный мультитаг. Этот объект находится в иерархии **AntaresController → MultiTagsCollector**

public static List<GameObject> FindObjectsFithMultitag(string multitag)

МОДУЛЬ СОХРАНЕНИЯ ОПИСАНИЯ ОБЪЕКТА.

Для вашего удобства, Antares Addon предлагает возможность сохранения описания любого объекта сцены или префаба в Проекте.
Описание хранится в переменной public string objectDescription компонента AntaresAddon. Если в настройках Antares Addon установлен флаг **Free Memory On Play** в режиме Play эта переменная очищается.



ANTARES BASKET

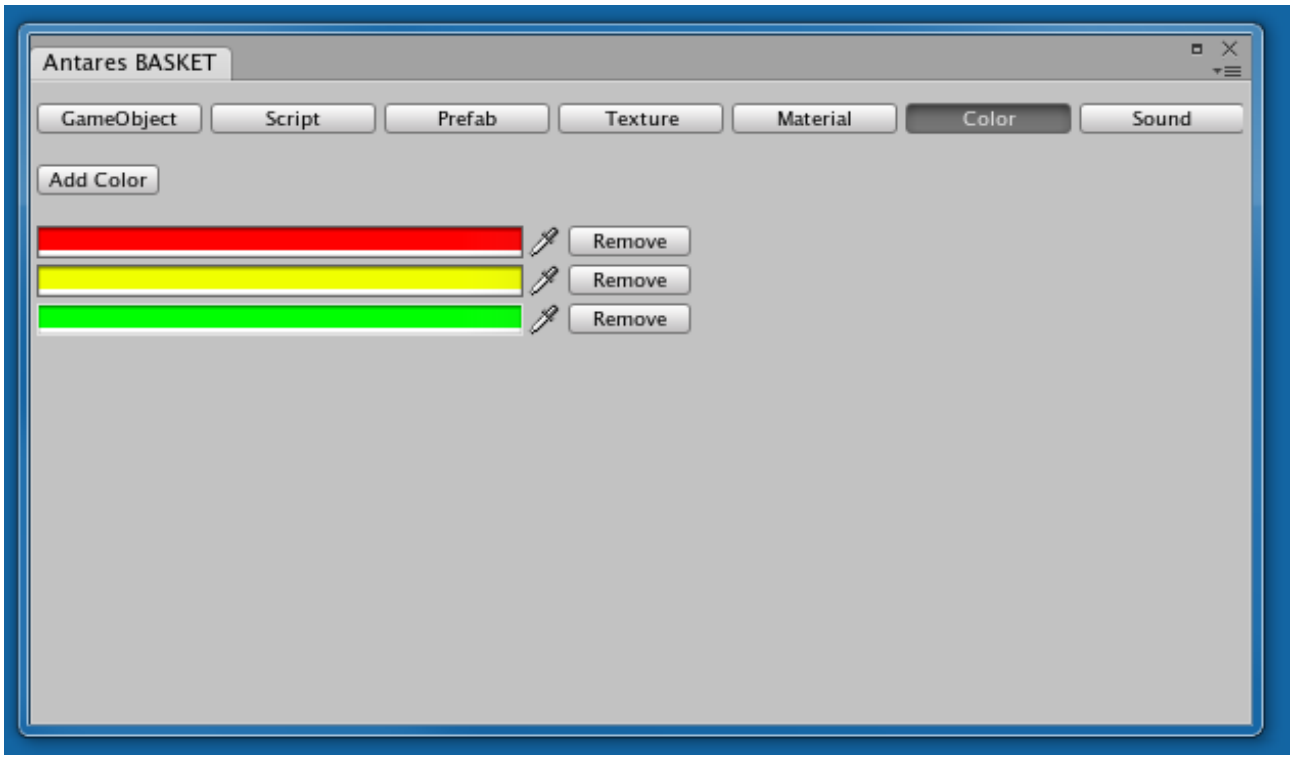
Antares Basket (*Alt + B (Windows) или Antares → Show BASKET*) наверное самый удобный инструмент для редактирования уровней в Больших Проектах из всех, вошедших в состав **Antares Addon.**
Корзина, представляет собой окно, принимающее Drag & Drop практически любые ассеты проекта. Она создана для того, чтобы вы имели возможность кинуть в неё нужные сейчас объекты и использовать их для быстрого и удобного изменения Сцены или настроек компонентов объектов.

Так же, **Antares Basket** умеет экспортировать любые типы ассетов в **AssetBundle** в два клика!

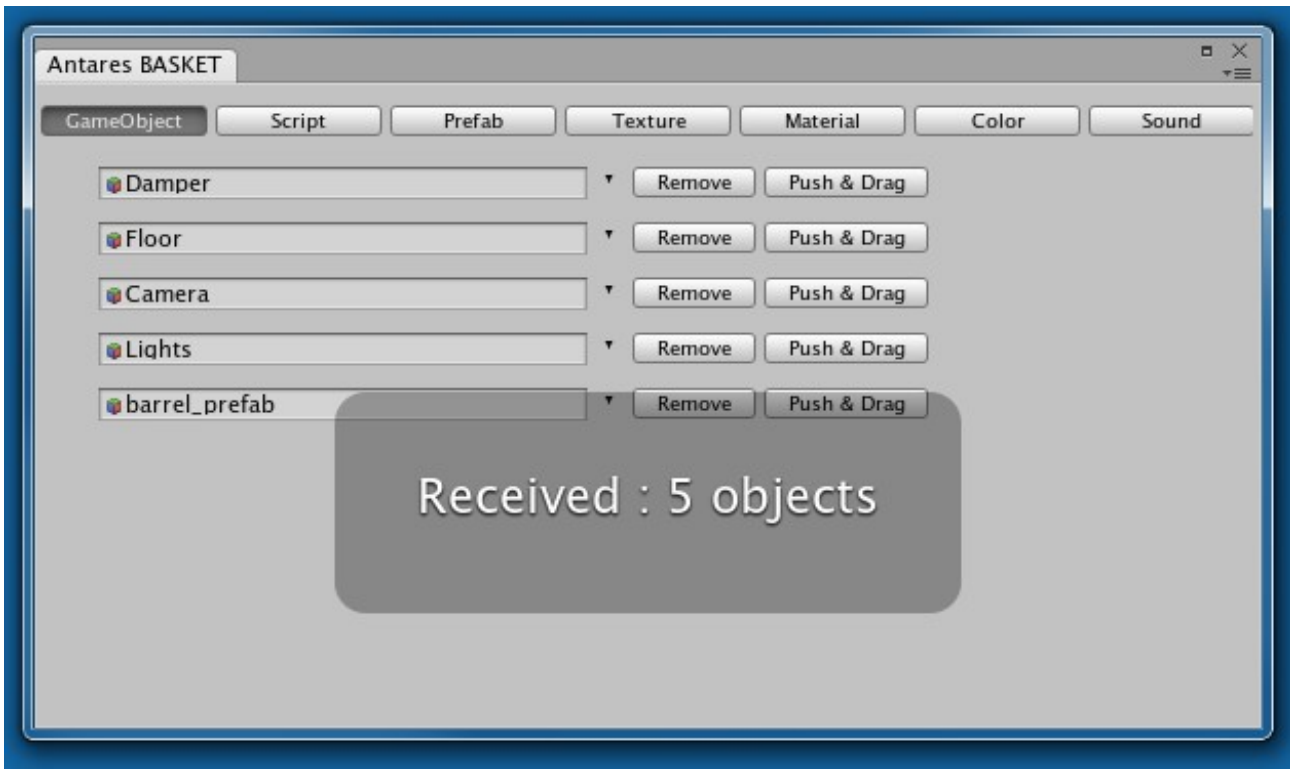
1. **Пример :** Вы редактируете игровой уровень. У вас Большой Проект и сотни или даже тысячи объектов в Project View. Искать нужные ассеты иногда становится серьёзным испытанием. Предположим, что в данный момент вам нужны только 5 или 15 префабов, чтобы расположить на игровом уровне

небольшую деревеньку.
Вы накидываете в окно Antares Basket нужные ассеты, находите место, где будет располагаться ваша деревенька и просто прямо из окна, перетаскиваете объекты в Сцену.

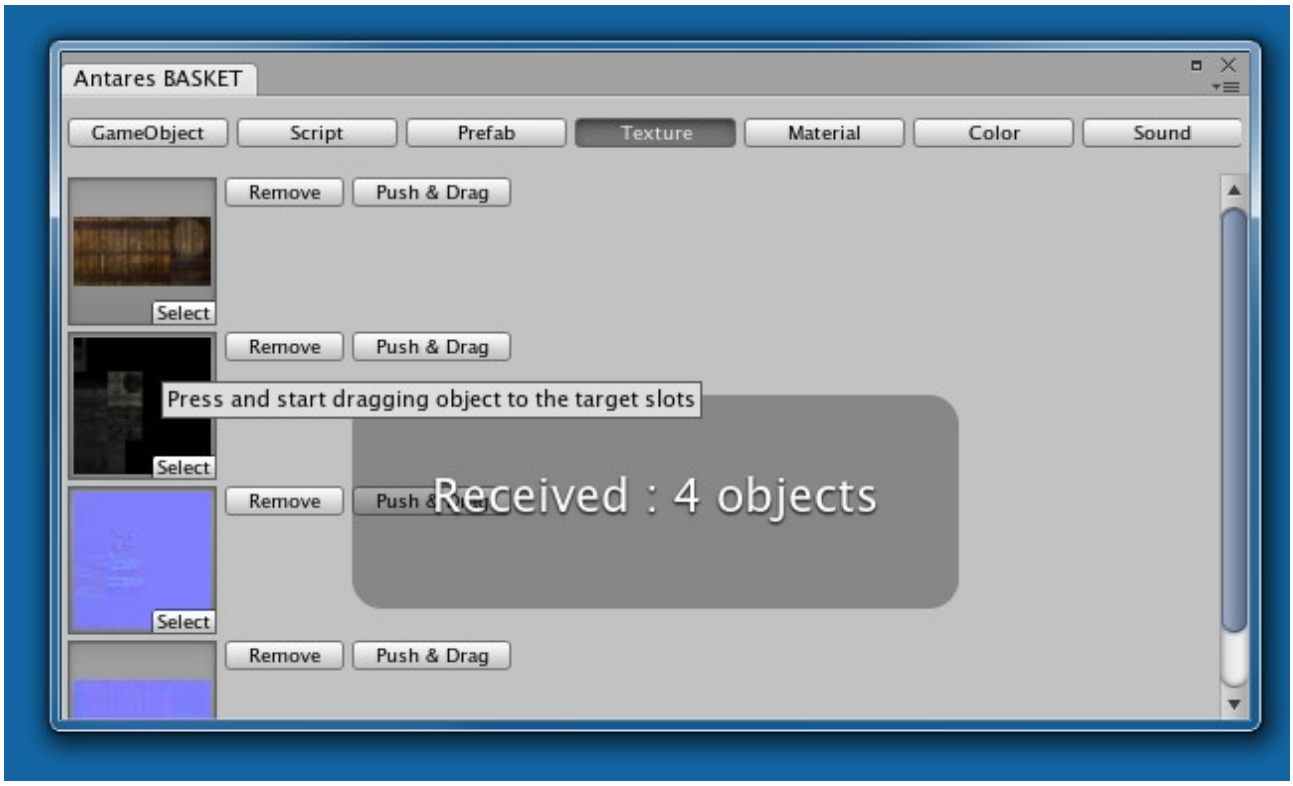
ПРИМЕЧАНИЕ : *Тут вам сильно помогут кнопки* ***Set Position*** *и* ***Make Brush!***
2. **Пример :** Вы хотите поменять цвет на 10 материалах. Пипетка **Unity** это хорошо, но вкладка Antares Basket → Color ещё лучше! Здесь вы можете создать свою палитру цветов и брать их по мере необходимости.



• ***Перетаскиваемые ассеты должны быть одного типа.***
◦ Только Модели



- **Только Текстуры**

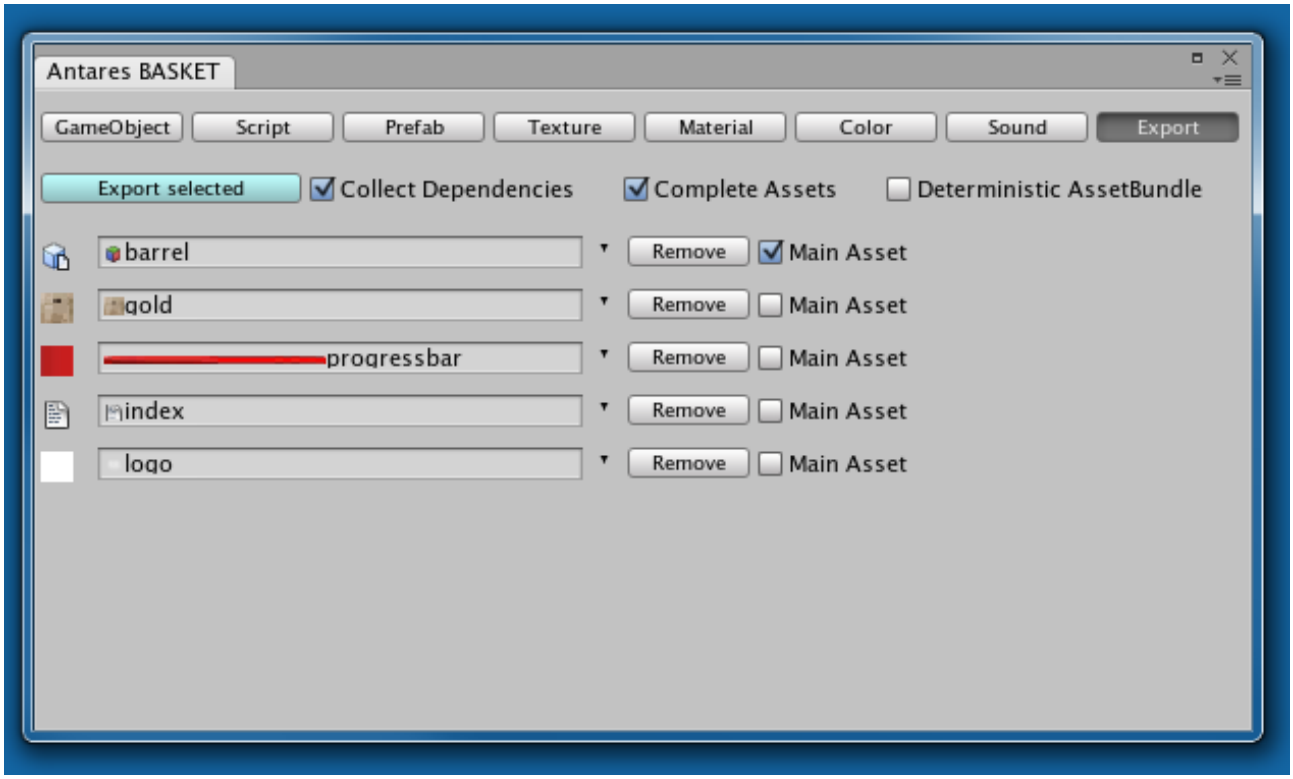


◦ **Только Материалы**
◦ пр.
• Ассеты автоматически раскладываются по вкладкам, соответственно их типам.
• Вкладка автоматически активируется. Вам не нужно выбирать, куда именно перетягивать ассеты. Просто киньте их в окно!
• Вкладки и объекты сохраняются после закрытия окна Корзины и повторного его

открытия.
ПРИМЕЧАНИЕ : *Список объектов очищается при выходе из Редактора* ***Unity***

ЭКСПОРТ В ФОРМАТ ASSETBUNDLE

Для того, чтобы подготовить пакет ассетов к экспорту, необходимо открыть вкладку **Export** и претащить в неё из окна Project View объекты, которые вы желаете експортировать в формат AssetBundle и выбрать один (!) Main Asset.
Чекбоксы настроек имеют всплывающие подсказки, объясняющие их значение.
Выберите настройки и нажмите **Export Selected**



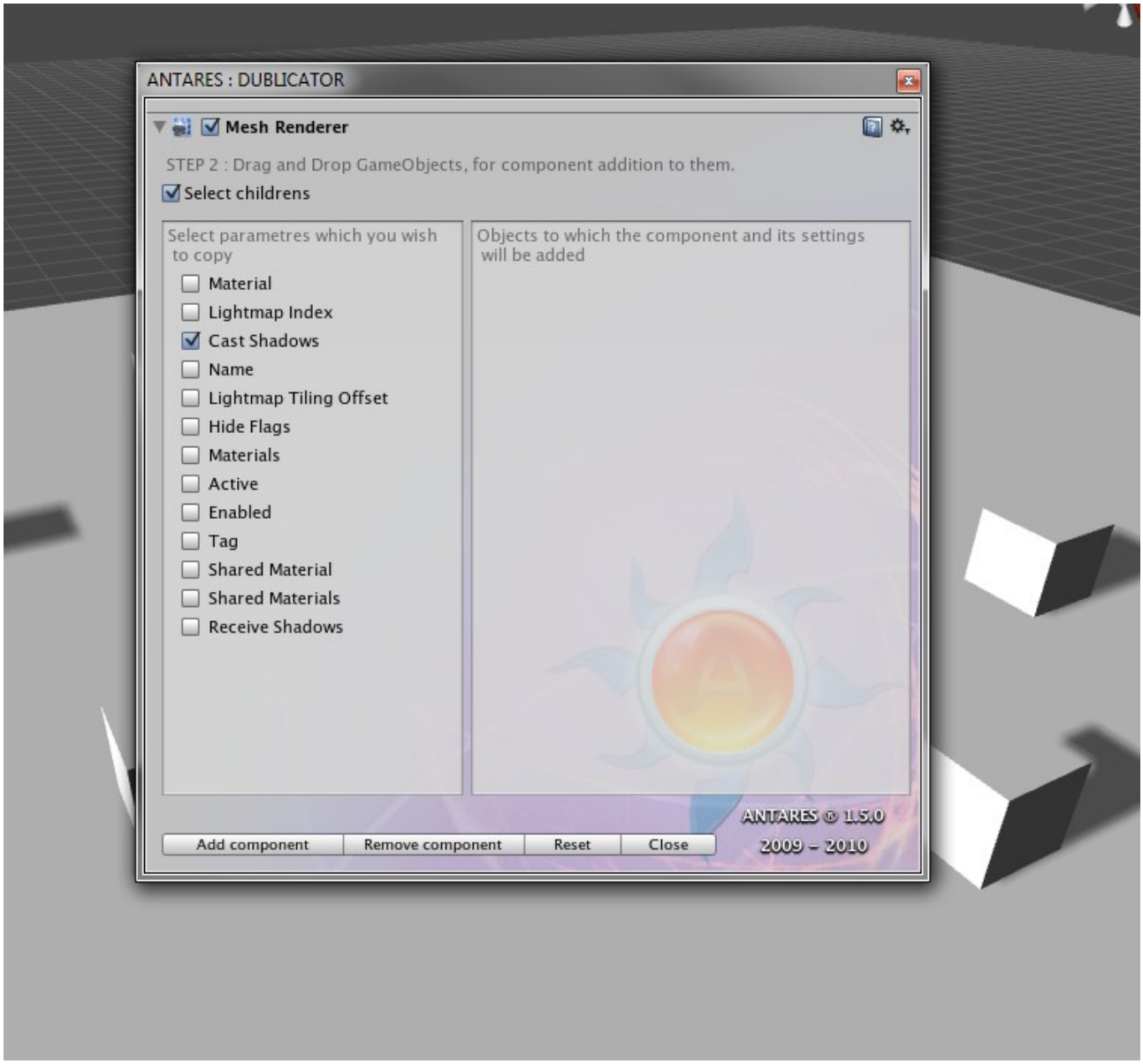
В открывшемся окне выберите имя файла и место его сохранения и нажмите Save.
Поздравляю! Ваш AssetBundle Создан!
ПРИМЕЧАНИЕ : *Экспорт в формат AssetBundle требует наличия лицензии* ***Unity PRO***

Antares DUBLICATOR

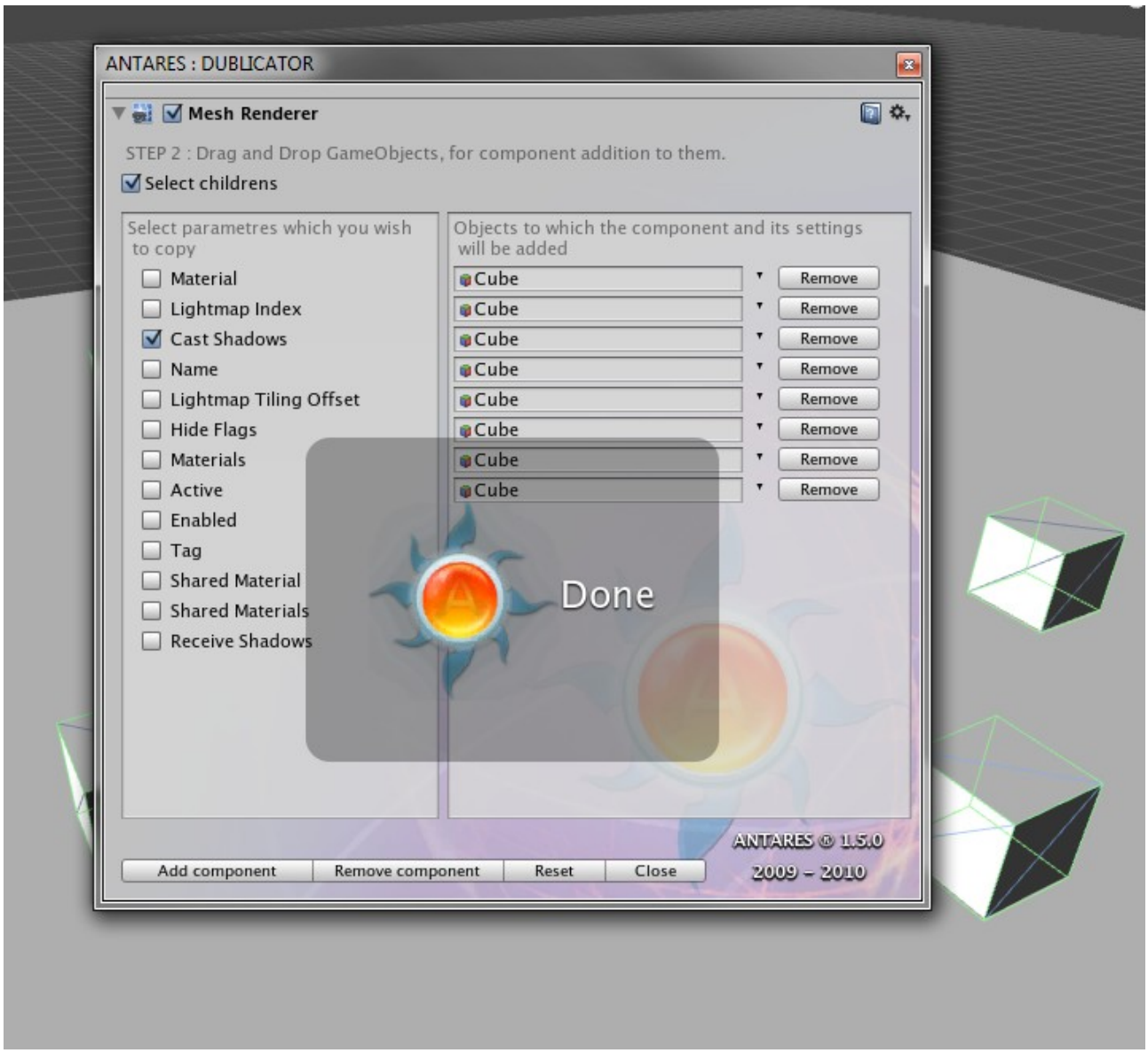
Antares DUBLICATOR это Wizard, созданный для простого и удобного копирования компонентов или настроек компонентов с одного GameObject на любое колличество других.

Команда меню Antares → DUBLICATOR открывает окно дубликатора. Копирование компонентов производится за два шага. К примеру, мы хотим отключить Cast Shadow на большом колличестве объектов.

Шаг первый : Выберите объект, с которого мы будем дублировать настройки. Отключите на его Mesh Renderer параметр Cast Shadow. Перетяните компонент (Component) Mesh Renderer в окно Дубликатора.
Вы увидите два поля, в левом поле будут отображены все переменные (variables, fields) Компонента, доступные для копирования.
Отмечаем Cast Shadow в левом окне.



Шаг второй : перетягиваем в окно объекты, на которые мы хотим скопировать настройки.



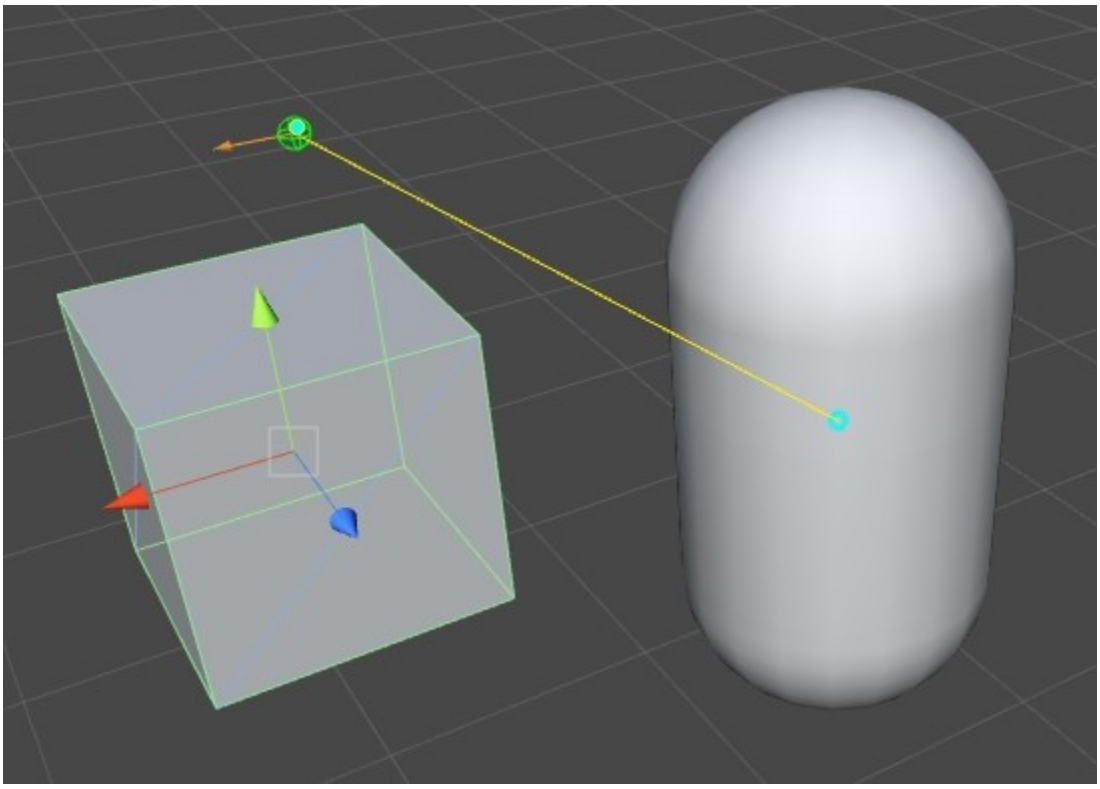
Нажимаем кнопку ”Add component” и видим, что тени от всех объектов исчезли.

TIPS: *Вы так же можете удалить любые компоненты с выбранных объектов или добавить компоненты к тем объектам, которые их не имеют.*

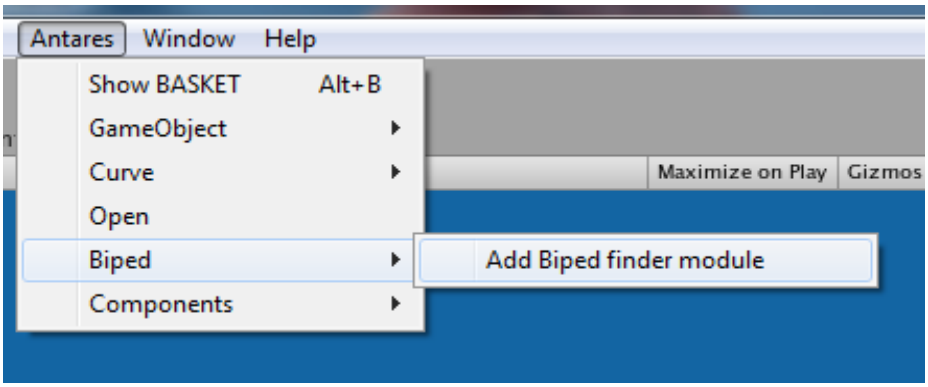
ВИЗУАЛИЗАЦИЯ ДЖОИНТОВ (JOINTS)

Для вашего удобства, Antares автоматически визуализирует связи джоинтов. На данный момент становятся видимыми :

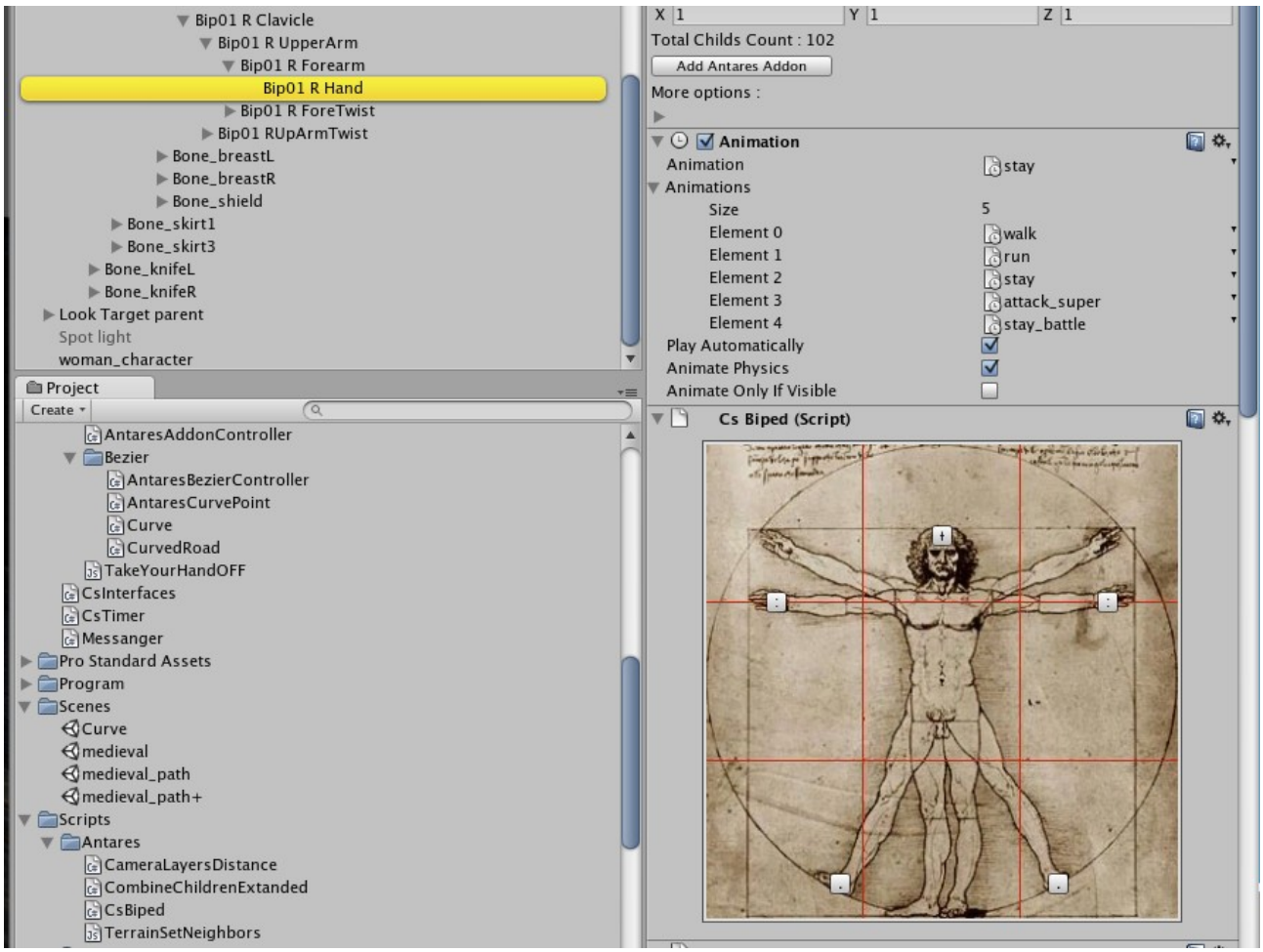
4. Hinge Joint
5. Fixed Joint



ИНСТРУМЕНТЫ : BIPED (Tools : Biped)

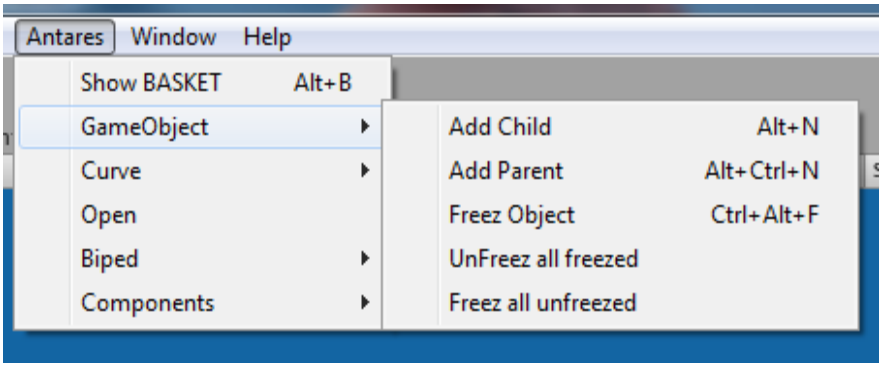


Antares → Biped → Add Biped Finder module
Эта команда добавляет к вашему персонажу модель поиска частей тела Бипеда.
Бипеды имеют сложную иерархию и найти в ней палец или ногу иногда непростая задача.
Этот модуль имеет только базовый aetrwbjyfk и расчитан на то, что вы сами отредактируете его под свои нужды.
В базовой реализации он ищет 5 частей тела :
1. Голова
2. Кисть левой руки
3. Кисть правой руки
4. Левую ступню
5. Правую ступню
Найденные объекты становятся легкодоступны по нажатию соответствующих кнопок модуля

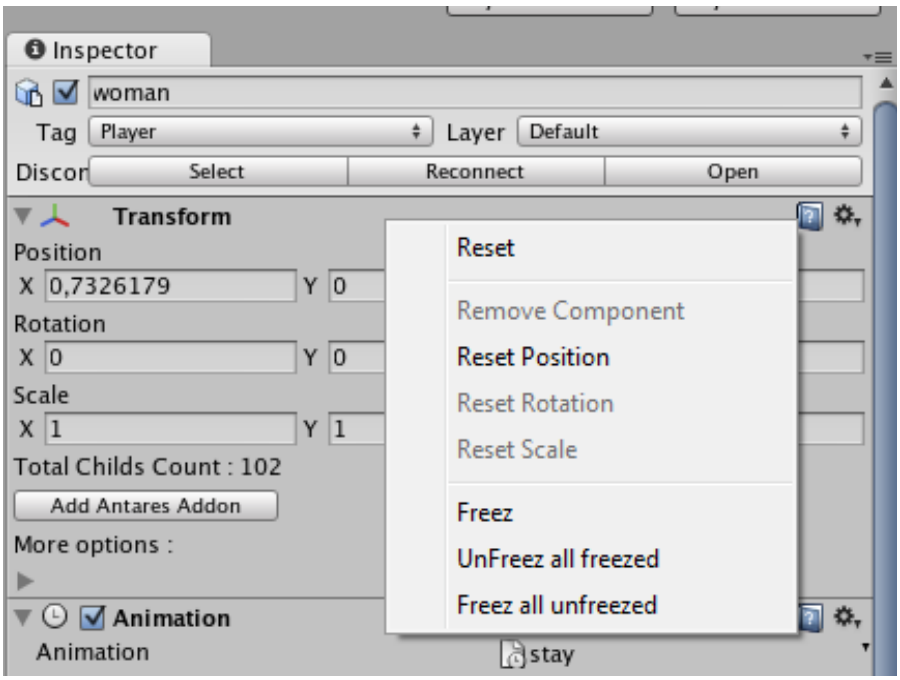


ИНСТРУМЕНТЫ : GameObject

Antares → GameObject



ПРИМЕЧАНИЕ : ***Часть функций этой вкладки доступна при правом клике на Инспекторе*** *Transform.*



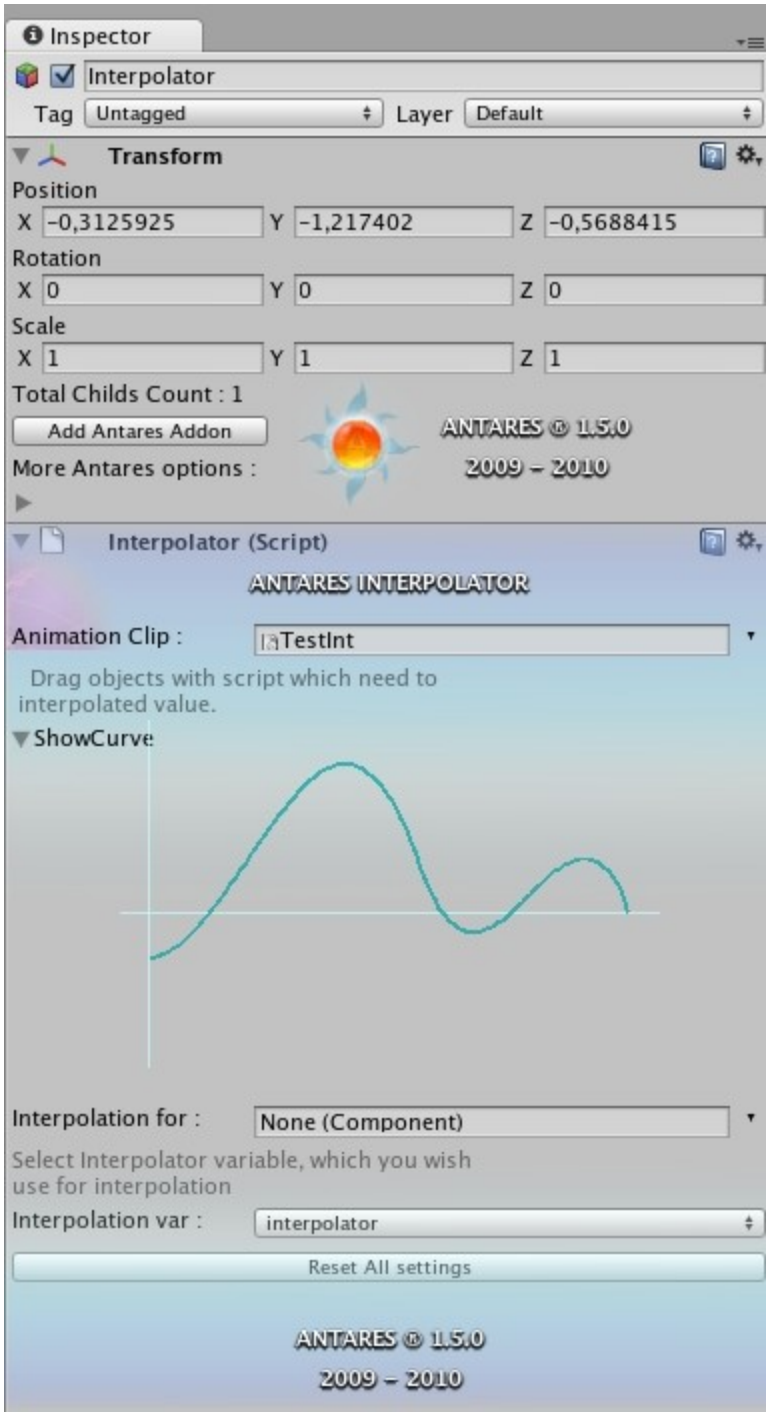
1. ***AddChild :*** *Добавляет к выбранному объекту пустой GameObject*
2. **Add Parent :** Добавляет выделенные объекты к новому пустому GameObject
3. **Freez Object :** Замораживает выделенный объект, запрещая его редактирование в Редакторе **Unity**.
4. **UnFreez All freezed :** “Размораживает” все ≪замороженные≫ объекты в Сцене.
5. **Freez all unfreezed :** “Замораживает” ранее ≪размороженные≫ объекты.

ANTARES INTERPOLATOR

Скачать демо проект.

Посмотреть онлайн.

Antares Interpolator это компонент, дающий возможность интерполировать float значение при помощи анимационных кривых, созданных при помощи великолепного Редактора Анимаций Unity.



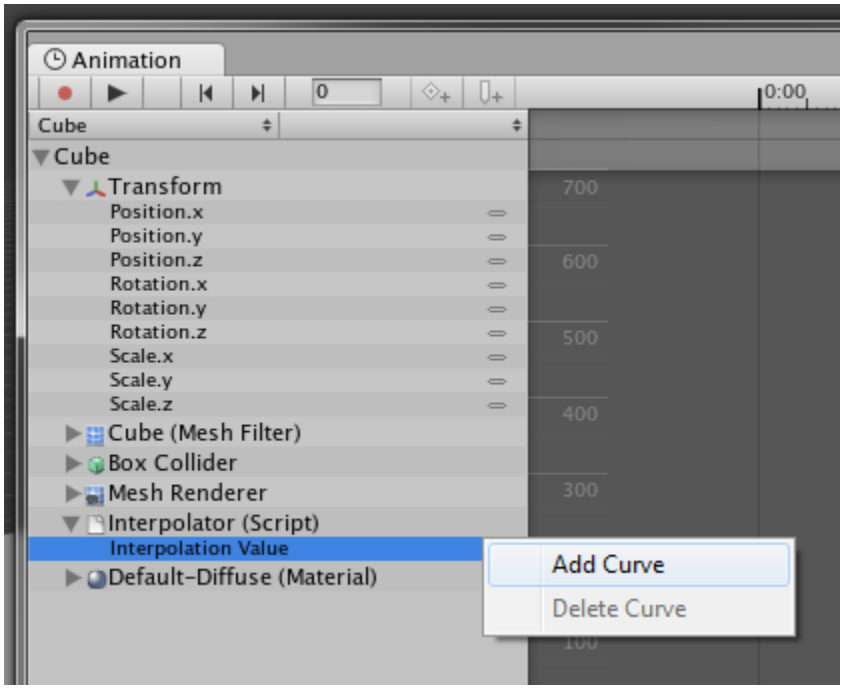
Для чего нужна интерполяция? К примеру, вы можете управлять скорость разгона и торможения автомобиля, при помощи этих кривых, визуально редактировать траекторию полёта ракеты или создать мерцание фонаря.

В чём же отличие от анимаций Unity ? Интерполятор позволяет обрабатывать любое количество кривых одновременно, не воспроизводя анимацию, а лишь получая значение кривой.

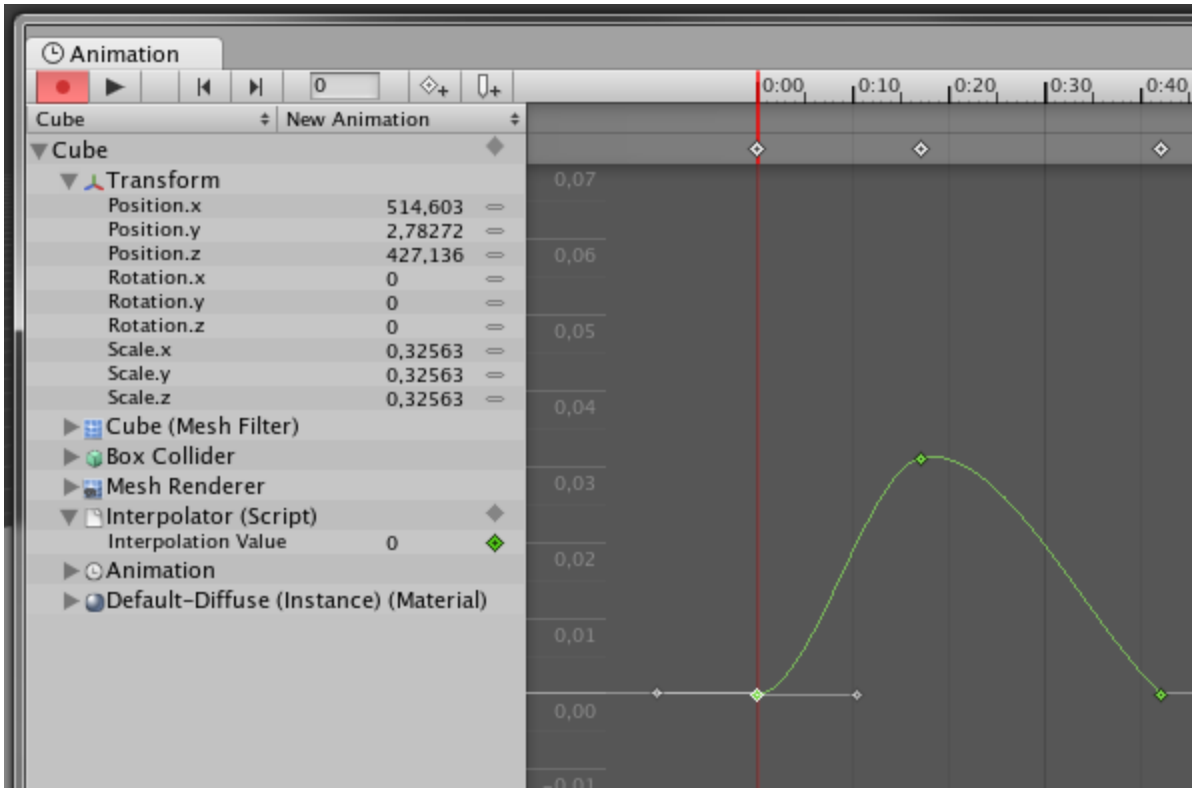
Внимание! На данный момент, на одном GameObject, может быть только один скрипт, работающий с Интерполятором, но Интерполяторов может быть сколь угодно много. То есть, этот скрипт, может работать сразу с несколькими кривыми одновременно.

Для того, чтобы получить интерполированное значение с кривой, вам необходимо проделать минимум действий :

- Добавить Antares Interpolator к GameObject.
- Создать или использовать сохранённый Animation Clip
  - Чтобы создать Animation Clip, выберем объект, содержащий Интерполятор.
  - Откроем окно Animation.
  - Найдём компонент Interpolator.
  - Добавим к его параметру Interpolation Value новую кривую (Curve)



  - Создадим кривую.



  - Закроем окно Редактора Анимаций.
- Анимация готова. Unity автоматически добавит компонент Animation к вашему

объекту. Его можно удалить, если вы не используете другие анимации на этом объекте. Или удалить из списка, только что созданную анимацию.

Для работы Интерполятора, необходим только файл анимации, компонент Animation не нужен и может быть удалён.

- ***TIPS :*** Созданная анимация может быть использована в любом другом проекте. Но, если вы используете её не в том проекте, в котором создали, у вас могут быть сложности с редактированием старой анимации.

- В скрипте, который будет получать и использовать интерполированные значения, нам нужно объявить переменную типа Interpolator :

```
public Interpolator interpolator, interpolatorAgain;
```

- Теперь просто перетянем GameObject, содержащий скрипт, который будет использовать интерполированные данные, в поле **Interpolation For**
- Ниже активируется поле **Interpolation Var**. Как вы видите, выше мы объявили две переменные типа Interpolator : interpolator и interpolatorAgain. В выпадающем списке **Interpolation Var**, вы можете выбрать желаемую переменную. Это сделано для того, чтобы один и тот же скрипт мог работать с любым желаемым количеством интерполируемых кривых одновременно.
- Осталось последнее — вызов Интерполятора и получение значения.

Компонент Интерполятор сам инициализирует переменные типа Interpolator, поэтому вам не нужно инициализировать их в Start или Awake. Всё что вам нужно — это вызвать метод интерполяции. Называются они по образу методов Unity Animation :

```
public void Reset()
public float Evaluate(float time)
public float EvaluateAutomatically()
```

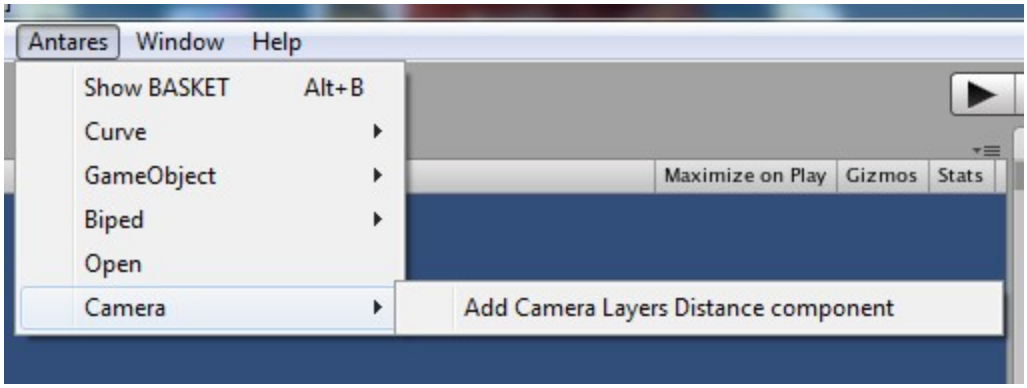
public void Reset() : Сбрасывает все параметры к исходным значениям. Если вызвать Reset из контекстного меню Редактора Unity, команда очистит переменные скрипта. Если из скрипта в runtime режиме, команда сбросит все параметры к состоянию старта.

public float Evaluate(float time) : Возвращает интерполированное значение. Входной параметр - «время» кривой (Curve) – оно зависит от длины анимационной кривой в секундах.

К примеру, если длина вашей кривой 5 секунд, то чтобы получить интерполированое значение кривой в её середине, необходимо вызвать float middle = Evaluate(0.25f);

public float EvaluateAutomatically() : автоматически воспроизводит кривую, в каждом кадре возвращая её значение. Если достигнут конец кривой, воспроизведение продолжается с начала.

ИНСТРУМЕНТЫ : КАМЕРА



Natares → Camera → Add Camera Layers Distance component

Эта команда добавляет к выбраной камере, компонент, позволяющий легко и визуально настраивать дистанцию видимости камеры в любом выбранном слое.
Это превосходный способ улучшить производительность вашего приложения.
Пример : Мелкие и несущественные объекты могут находиться в слое «Garbage» а важные и большие в слое «1000 meters» и камера не будет рендерить объекты в данных слоях на дистанциях, превышающих заданные значения.

Antares.dll
(не включено в основной паккадж)

Manager
Free for non-commercial use.
neodrop@unity3d.ru

Antares.Manager Class allows building programs based completely on events.
Any component (Event Listener) waiting for a certain Event, is registered in Antares.Manager, by giving self-reference (as an object), priority index in sequence and string parameters — names of functions in which the component will wait for Invoke call or Message (depending on the type of the Event determined by the Event Sender).
Here's a typical example of registration:

```
void Awake()
{
        Antares.Manager.RegisterComponent(this, 0, “MouseButtonDown”, “MouseButtonUp”);
        или
        Antares.Manager.RegisterComponent(this, 0, MouseButtonDown, MouseButtonUp);
        //(in such case, function — listener must have zero arguments)
}

void MouseButtonDown()
{
        Debug.Log(“I have event about mouse button down”);
}

void MouseButtonUp()
```

```
{
        Debug.Log("I have event about mouse button up");
}
```

The priority index allows adjusting the sequence of Events recived by Listeners.
When necessary, Listener may deregister and stop receiving messages.

```
Antares.Manager.UnRegisterComponent(this, "MouseButtonUp");
```

Also, you may register several types of delegates:

Method of delegates with zero arguments:

```
void OnMouseDown()
{

}
```

The method of delegates with the parameter of object type as an argument:

```
void OnMouseDown(object btn)
{
        switch((string)btn)
        {
                case "left" :
                break;
                case "middle" :
                break;
                case "right" :
                break;
        }
}
```

Or :

```
void OnMouseDown(object btn)
{
        switch((int)btn)
        {
                case 0 :
                break;
                case 1 :
                break;
                case 2 :
                break;
        }
}
```

Tips : As an argument, you may transfer an array of object type by extending considerably the possibilities of the method - listener.

Also, you may transfer the own type of Delegate, with any desirable set
of arguments :

```
delegate void TestDelegate(string a, Vector3 b, Color c);

void TestCallingMethod(string a, Vector3 b, Color c)
{
	Debug.Log("String = " + a + " , Vector = " + b + " and Color = " + c);
}

void Awake()
{
	thisScript = this;
	Antares.Manager.RegisterComponent(this, 0, new TestDelegate(TestCallingMethod));
	Antares.Manager.EventMessageCustom("TestCallingMethod", "Test", Vector3.forward,
Color.cyan);
}
```

IMPORTANT! When using Custom Delegates, you must call Events only by methods that have the word
Custom in their names (otherwise, they will be called but will not receive their arguments):

Antares.Manager.EventMessageCustom
Antares.Manager.EventMessageExcludeCustom

Control of Events.

Any script controlling Event in the program, or generating it independently, may call easily Antares.Manager and notify all Listeners on the Event occurred.

Typical example :

```
void OnMouseDown()
{
	Antares.Manager.Event(“MouseButtonDown”);
}
```

All Listeners, in the order of the preset priority, will receive immediately Event in the function MouseButtonDown.

Events may be called for all registered listeners or only for preset types:

```
Antares.Manager.Event(“MouseButtonDown”,  typeof (ButtonsWaiter), typeof(Soldiers));
```

When it is necessary to exclude a Listener or a group of Listeners from the list of Event Recepients, the following function is used :

```
public static void EventExclude(string eventName, params object[] excludeComponents)
```
где params object[] excludeComponents – list of Listeners who will not receive an Event.

Transfer of parameters at transferring an Event.

When necessary, together with an Event, parameters may be transferred in the same manner as it occurs at using GameObject.SendMessage() method.

```
Antares.Manager.EventMessage("MouseButtonDown",  true, typeof (ButtonsWaiter),
typeof(Soldiers));
```
or
```
Antares.Manager.EventMessage("MouseButtonDown",  soldiersArray);
```
(where SoldiersArray, for example, has type ArrayList)

```
or Antares.Manager.EventMessageCustom("MouseButtonDown",  soldiersArray, 100, "Attack!");
```

Blocking Events.

Upon the wish of the program's author, the Event may be blocked for all recipients:

```
Antares.BlockEvents("MouseButtonDown", "MouseButtonUp" );
```

or unblocked:

```
Antares.UnBlockEvents("MouseButtonDown", "MouseButtonUp" );
```

Debugging.

Method Antares.ShowAllEventsList(true); it returns back the massive string[] containing all registered Events. At parameter debugLog == true, all Events are displayed to the Unity console.

List of all functions:

Registering Listeners and Events

```
public static void RegisterComponent(object component, int priority, params AntaresDelegateNull[]
customDelegate);
public static void RegisterComponent(object component, int priority, params Delegate[]
userDelegate);
public static void RegisterComponent(object component, int priority, params DelegateCustom[]
customDelegate);
public static void RegisterComponent(object component, int priority, params string[] eventsName);
public static void UnregisterComponent( object component, params string[] eventsName)
public static void UnregisterEvent(string eventName)
```

Calling Events

```
public static void Event(string eventName, params System.Type[] typs)
public static void EventExclude(string eventName, params object[] excludeComponents)

public static void EventMessage(string eventName, object value);
public static void EventMessage(string eventName, object value, params Type[] typs);
public static void EventMessageCustom(string eventName, params object[] value);
public static void EventMessageExclude(string eventName, object value, params object[]
excludeComponents);
public static void EventMessageExcludeCustom(string eventName, object excludeComponent,
params object[] value);
```

Control of Events Management.

```
public static void BlockEvents(params string[] events)
public static void UnBlockEvents(params string[] events)

public static string[] ShowAllEventsList(bool debugLog)
```

Additional class AGUI

Class Antares.AGUI recognize the Events of entering and exiting of the mouse cursor within a Rect.

public static bool MouseOverGUI(Rect rect, ref bool lastOverCondition, string enterGUIMessage, string exitGUIMessage, params Vector3[] inputMousePosition)

Rect rect – The parameter Rect contains a Rect of the GUI element.
ref bool lastOverCondition – bool variable, mandatory for each checked GUI element. At the start of the element existence, it must be
set in the status false. Namely it allows determining
the Event of entry/exit within the limits of GUI element.
The class processes any number of GUI elements simultaneously and generates the Event of entry/exit within the limits of GUI only once per frame.
Thus, any random number of elements (windows, buttons, etc.) may be processed within one frame and the Event of entry/exit of the cursor will be sent to Listeners only once. That saves the programmer the trouble of checking each element independently and many times reduces the quantity of necessary code.

string enterGUIMessage, string exitGUIMessage — optional parameters of an Event. If they are specified, the class will generate independently and send the Event to all Listeners.If not specified, it will just return bool variable,
designating the cursor position within the limits of the checked Rect.

The returned value will be generated always, independent of whether the optional parameters string enterGUIMessage, string exitGUIMessage are specified.

params Vector3[] inputMousePosition – optional parameter of the mouse position. May be not transferred.
In such case, the class will obtain the mouse position independently.

public static int GetUIDforWindow() - generates unique int number for GUI windows.

List of functions:

public static bool MouseOverGUI(Rect rect, ref bool lastOverCondition, string enterGUIMessage, string exitGUIMessage, params Vector3[] inputMousePosition)
public static bool MouseOverGUI(Rect rect, Vector3 mousePosition)
public static bool MouseOverGUI(Rect rect)
public static int GetUIDforWindow()

Additional class : BinarySave

Class of binary serialization / deserialization of data.
It operates both on Windows and on MacOS X

Binary data serialization uses smaller file size than XML serialization and does it much faster.
But it has a number of peculiarities.

Binary serialization in Unity is a bit simpler and simultaneously a bit more complicated than XML serialization.
The problem is that the majority of structures and classes of Unity have no Serializable flag and in order to save, for example,
Vector3, it is necessary to save not the object itself (Vector3) but its variables X, Y, Z.

And here we have the main suprise: due to peculiarities of MONO, after each compilation of the project or any of its scripts, all classes of the Project get new identifiers (not GUID) and binary files,
recorded by your program during earlier startups, become unreadable for the program.
The solution is simple but for those who do not know C# well, it is rather unexpected.
It is necessary to pack the class of data to be saved into .NET DLL
After that you may change anything you want both in the program itself and in this DLL, and you will have no problems with data deserialization.

By using this class and the selected traffic, I easily save and upload data on 5000 cars, without any loss in the application productivity.
So :

Mandatory function of indication of the folder with which class BinarySave will operate:

```
public static void SetCurrentFolder(string folder)
```

The path is specified in relation to the file executed. When working in Unity Editor, this is Assets folder.
That is, if you call SetCurrentFolder("MyData") method, when working in Editor, all other methods will look for MyData folder in Assets folder.
The operation of this function is backed up completely by the method public static void CreateFolder(string folder)

```
public static void CreateFolder(string folder)
```

It creates the folder according to the path specified, in relation to the executed program file.

```
public static void CreateFolderWithFullPath(string fullPath)
```

It creates the folder according to the absolute path.

```
public static void DeleteFolderWithFullPath(string fullPath, bool includeSubfolders)
```

It deletes the folder according to the absolute path. The variant for the relative patrh is not realized.

```
public static string GetCurrentFolder()
```

Returns the current “currentFolder”

public static void Save(object obj, string fileName, string fileExtansion)

Serializes and saves your data in currentFolder.
Example:
Antares.BinarySave.Save(obj, “Data”, “.sav”);

public static object Load(string fileName, string fileExtansion)

It deserializes the file saved.
Example:
MySaveClass[] saver = (saver[])Antares.BinarySave.Load(obj, “Data”, “.sav”);

ANTARES ASYNC API

Классы Antares.Async созданы для упрощения загрузки AssetBundles и WWW объектов. Главная их цель — следить и предотвращать попытки повторной загрузки имеющихся AssetBundles, упростить повторные запросы к объектам и предоставить статистические данные о загруженных AssetBundles, а так же методы управления памятью.

ANTARES.ASINC.LOADEDOBJECT :

Класс, сожержащий загруженный AssetBundle и реализующий методы управления и доступа к загруженным ассетам и их содержимому.

Переменные :
public bool done : Отмечает, что AssetBundle загружен.
public System.Type objectType : Тип объекта, загружаемого из AssetBundle
public float loadedTime : Время начала загрузки
public float loadingTime : Время, затраченное на загрузку
public List<Loader.AsyncLoaderDelegate> waiter : Список делегированных методов, зарегистрированных для ожидания конца загрузки этого объекта.
public string path : Путь загрузки
public float wwwProgress : Прогресс загрузки
public WWW www : Объект класса WWW, осуществляющий загрузку.

Методы :
(Перечислены только те методы, ипользование которых допускается.)
public void UnloadAsset(bool unloadAll) : Выгружает AssetBundle из памяти.
public string GetObjectName() : Возвращает имя загруженного объекта
public Object Instantiate() : Инстансирует объект. Если вы хотите использовать встроенный контроль за копиями объекта (GetListOfIstancies()) используйте этот метод, для инстансирования, вместо UnityEngine.Instantiate()
public List<Object> GetListOfIstancies() : Возвращает лист всех копий загруженного из AssetBundle объекта, если он был инстансирован при помощи LoadedObject.Instantiate();
public Object GetAsset() : Возвращает AssetBundle.

ANTARES.ASINC.LODER :

public static void LoadObject<T>(string path, string objectName, AsyncLoaderDelegate loadCallback, AsyncProgressDelegate loadProgressCallback, int priority)

<T> : тип объекта, загружаемого из AssetBundle. Это может быть любой ассет, такой как :

4. Texture
5. Material
6. GUISkin
7. GameObject
8. прочее

string path : Путь к загружаемому AssetBundle.
string objectName : Имя объекта, загружаемого из AssetBundle.
AsyncLoaderDelegate loadCallback : делегированный метод, в который будет произведён вызов при завершении загрузки.
AsyncProgressDelegate loadProgressCallback : делегированный метод, в который будет производится вызов в каждом кадре. Служит для получения прогресса загрузки.
Может быть равен null.

Если AssetBundle уже был ранее загружен, Antares.dll произведёт вызов в loadCallback и передаст загруженный объект.

Типичное использование :

```
void Awake()
{
      Antares.Async.Loader.LoadObject<GameObject>("http://www.unity3d.ru/composition/Giraf_bundle.unity3d","model",OnLoad, OnProgress, 0);
}

void OnLoad(Antares.Async.LoadedObject obj)
{
        obj.Instantiate();
}

 private float loadingProgress;
void OnProgress(float progress)
{
     loadingProgress = progress;
}
```

ВНИМАНИЕ! При невозможности загрузки AssetBundle или невозможности загрузки заданного объекта из AssetBundle, вызов в делегированный метод loadCallback будет произведён всё равно. Рекомендую проверять переданный экземпляр LoadedObject.www.AssetBundle на null перед инстансированием.

public static void LoadWWW(string path, AsyncLoaderDelegate loadCallback, AsyncProgressDelegate loadProgressCallback)

Аналогичный метод, позволяющий загружать объекты WWW таким же образом, как и

AssetBundles
После завершения загрузки, Antares.dll произведёт вызов делегированного метода loadCallback и передаст ему объект типа LoadedObject, откуда вы можете взять объект www и использовать его по своему усмотрению.

public static void UnloadAll() : Выгружает все загруженные AssetBundles из памяти.

Created by Andrey Paramonov (aka Neodrop).
19.01.2010, Saint-Petersburg, Russia

neodrop@unity3d.ru

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

Фокус на объекте : Работает так же как кнопка F – фокусирует камеру Сцены на выбранном объекте, но по нажатию колёсика мыши.

Copy Animation Clip : Вы можете скопировать анимационный клип для дальнейшего его редактирования, просто выбрав его в Project View и выбрав правой кнопкой мыши команду Antares/Copy Animation Clip

Hide/Unhide in Hierarchy View : Вы можете скрыть любой объект из окна Hierarchy View, просто нажав кнопку Hide In Hierarchy в меню Инспектора Трансформ (в скрытом списке more Antares options).

АВТОРЫ

1. **Евгений Плешков** (aka **Ferz**) (www.unity3d.ru, fer-7@ya.ru) :
    1. Random Position
    2. Random Angles
    3. Set angles to 0
    4. MakeBrush (75%)

2. **Андрей Пахомов** (aka **PAX**) (www.unity3d.ru, pax83@mail.ru) :
    Antares DUBLICATOR (reflection code).

3. **Александр Хомяк** (aka **Norgen**) (www.unity3d.ru, rivia@mail.ru) :
    Art.

4. **Слава Панкратов** (aka **Gnoblin**) (www.unity3d.ru, gnoblin@gmail.com) :
    Перевод на английский.

5. http://www.reignofsteel.net :
    1. Add Child
    2. Add Parent
6. **Андрей Парамонов** (aka **Neodrop**) (www.unity3d.ru, neodrop@unity3d.ru)

∞ Всё остальное.

ПОМОЧЬ ПРОЕКТУ

Помочь проекту вы можете тут.