No Man's Sky – процедурно генерируемый контент

*Кириченко В.В. – Перевод статьи «No Man's Sky – procedural content» (http://3dgamedevblog.com/wordpress/?p=836).

No Man's Sky – игра об исследовании космоса, где используется технология процедурной генерации для создания игрового окружения и ассетов (текстур, моделей, поверхностей и т. д.). Я был очень взволнован еще с того момента, когда эту игру анонсировали в 2013 году – не только по причине самой игры, но и из-за прекрасной возможности начать свое исследование файлов игры, чтобы разобраться, как это работает. Собственно, No Man's Sky оказалась одной из самый спорных среди вообще всех выпущенных игр, но ее внутренний механизм все еще представляет неподдельный интерес.

Каждый, кто устанавливает эту игру, может легко обнаружить, что ее размер крайне мал для ее объемности, и это правда. Однако главной причиной этого является тот факт, что игра, используя процедурную генерацию, работает с совершенно малым набором ассетов для создания практически сотен вариантов. Сейчас я хочу рассмотреть контент, связанный с игровыми 3D-моделями, потому что это было всегда мне интересно. Я также разделю статью на три основные части: 1) Геометрию; 2) Текстуры и 3) Анимацию.

ГЕОМЕТРИЯ

Итак, внутри игровых файлов есть чисто геометрические (вертекс, индекс-буферы и пр.), которые хранятся в хранятся в файлах с расширением “.GEOMETRY.MBIN”. С помощью этих файлов можно создать очень простые парсеры, способные перенести эту геометрию в ПО для моделлинга. НО этого файла не достаточно самого по себе. В действительности файл геометрии ведет себя как контейнер для чисто геометрических данных.

Игра загружает ассеты путем графа сцены. Это значит, что все ассеты моделей определены как отдельные файлы сцены с кастомной иерархией объектов, частями полигональных сеток, различными типами объектов (стыки, свет, области взаимодействия, коллизии и другие файлы сцены) и т. д. Этот тип информации хранится в “.SCENE.MBIN”-файлах. Они и есть настоящие дескрипторы для каждой сцены, и обычно эти файлы относятся к одному геометрическому контейнеру, откуда все части полигональной сетки

конкретной сцены получают надлежащую им геометрическую информацию.

На этом этапе нет ничего нового. Это весьма обычное явление для многих современных игр. Что отличается в NMS (по крайней мере, я не встречал такого ранее никогда), так это то, что внутри файла сцены нет ни одного вручную созданного существа, которое может появиться в игре в связи с определенными обстоятельствами, зато там есть процедурная генерация.

Для того, что было легко понять, как это выглядит, я прикреплю несколько картинок созданного мной Model Viewer'a для NMS.

Модель Трицератопса

Как вы можете видеть, на первый взгляд сцена выглядит как полнейший хаос, и вы даже не можете сказать, что вы видите перед собой. В действительности вы и не видите ничего такого, что может появиться как-либо в игре.

Ключом к этой загадке являются настоящие названия объектов. Легко заметить, что есть некоторая связь между ними, и игра должна каким-то образом знать, как объединять эти части, и как исключать объединение с другими во время создания модели.

Поняв это, я начал искать другие файлы, способные справиться с этим типом организации сцены, и мне кажется, что именно “.DESCRIPTOR.MBIN”-файлы этим занимаются. Не все модели имеют такие файлы. Вообще, оказывается, что они есть только у процедурно генерируемых моделей, в то время как статические (такие, как модели космонавтов или некоторые крафт-материалы) их не имеют.

Когда я впервые стал изменять форматы файлов в NMS, я начал с изучения файлов-дескрипторов, но они сразу же мне показались абсурдными, и

я не смог разобраться, что они делают. После разбивки сцен на составляющие с учетом того, что я ищу, я вернулся к файлам-дескрипторам. Эти файлы очень сильно похожи на файлы сцены. У них тот же тип определения частей и пр., НО они могут относиться только либо к частям полигональной сетки, либо к целым файлам сцены. В общем, то, как это работает, выглядит таким образом:

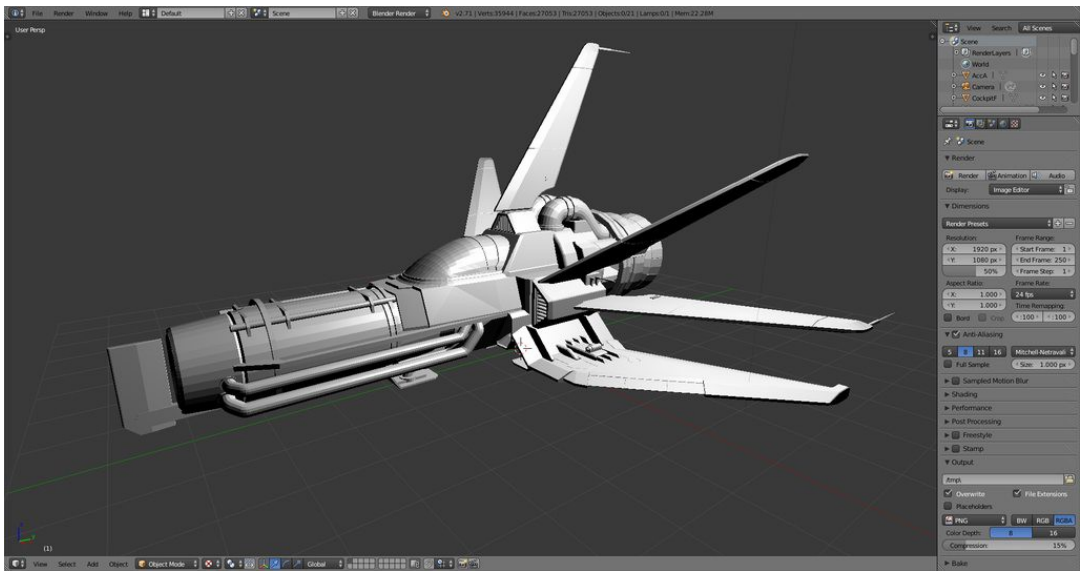
```
<Data template="TkModelDescriptorList">
 <Property name="List">
  <Property value="TkResourceDescriptorList.xml">
   <Property name="TypeId" value="_HEAD_" />
   <Property name="Descriptors">
    <Property value="TkResourceDescriptorData.xml">
     <Property name="Id" value="_HEAD_ALIEN" />
     <Property name="Name" value="_Head_Alien" />
     <Property name="ReferencePaths" />
     <Property name="Chance" value="0" />
     <Property name="Children">
      ......;
     </Property>
    </Property>
    <Property value="TkResourceDescriptorData.xml">
     <Property name="Id" value="_HEAD_DIPLO" />
     <Property name="Name" value="_Head_Diplo" />
     <Property name="ReferencePaths" />
     <Property name="Chance" value="0" />
     <Property name="Children">
      ......;
     </Property>
    </Property>
    <Property value="TkResourceDescriptorData.xml">
     <Property name="Id" value="_HEAD_HIPPO" />
     <Property name="Name" value="_Head_Hippo" />
     <Property name="ReferencePaths" />
     <Property name="Chance" value="0" />
     <Property name="Children">
      ......;
     </Property>
    </Property>
    <Property value="TkResourceDescriptorData.xml">
     <Property name="Id" value="_HEAD_RHINO" />
     <Property name="Name" value="_Head_Rhino" />
     <Property name="ReferencePaths" />
     <Property name="Chance" value="0" />
     <Property name="Children">
      ......;
     </Property>
    </Property>
    <Property value="TkResourceDescriptorData.xml">
     <Property name="Id" value="_HEAD_STEG" />
     <Property name="Name" value="_Head_Steg" />
     <Property name="ReferencePaths" />
     <Property name="Chance" value="0" />
```

```
<Property name="Children">
  .....;
 </Property>
</Property>
```

Процедура проходит так: есть главная часть, которая должна быть определена для чего-то. В данном случае это _HEAD_ (голова). Обычно, части, названные с подчеркиванием перед ними, означают, что они являются частью группы дескрипторов, и только один из них будет выбран для конечной модели. Теперь, как вы можете видеть, эта часть определяется как TkResourceDescriptorList. Эти элементы, чьи субэлементы (Children) участвуют в соревновании на выбор, содержат свойство «Descriptors». Все, что вам нужно сделать дальше, так это выбрать однин из субэлементов. Например, таким образом может быть выбрана модель головы. Для особой модели головы, что была выбрана, есть другой другое свойство «Descriptors», у которого наличествует свой список доступных опций, и вам снова надо выбрать один из них. И так далее.

После того, как вы сделаете так со всеми предметами в descriptor.mbin-файле, вы получите уникальную и законченную модель.

SHARKRIG



Боевой космический корабль – взятый из NMS Model Viewer.

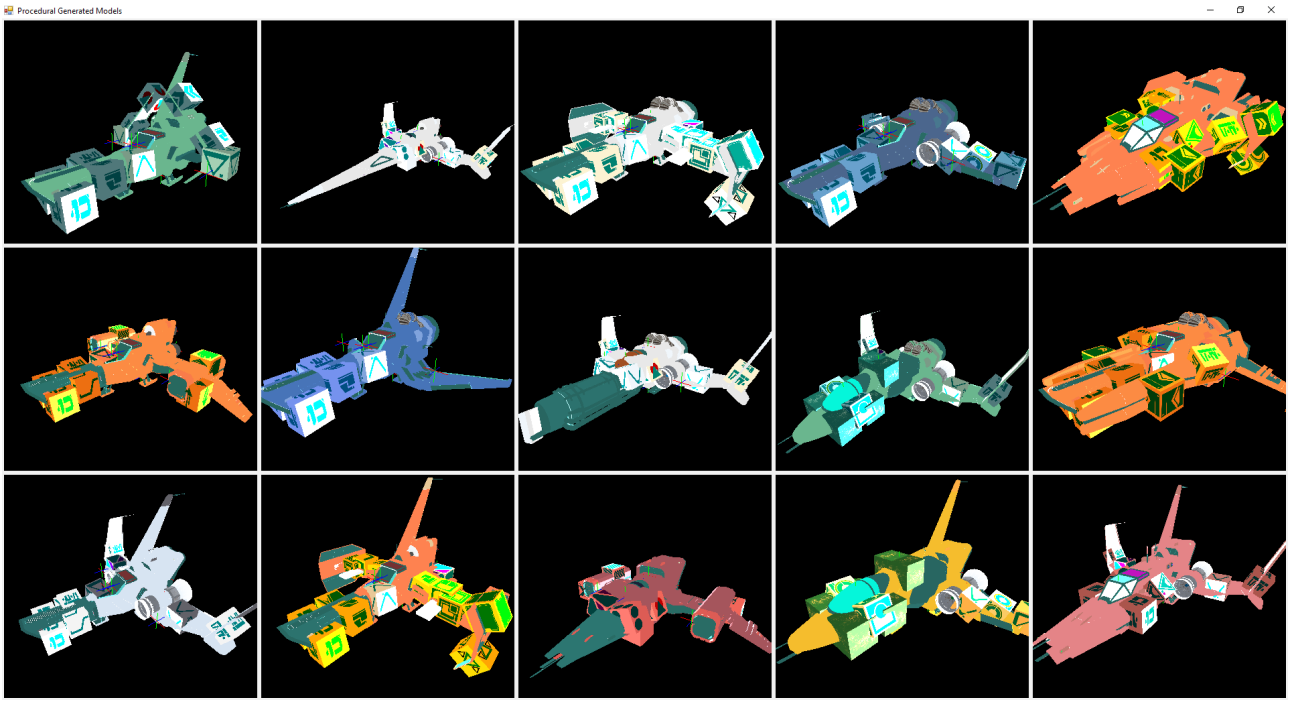
Охранники

Вся процедура – это настоящее дерево пересечений. Корень дерева – законченная сцена со всеми деталями, и в конечном результате это уникальная ветвь дерева, которое демонстрирует модели только с необходимыми частями.

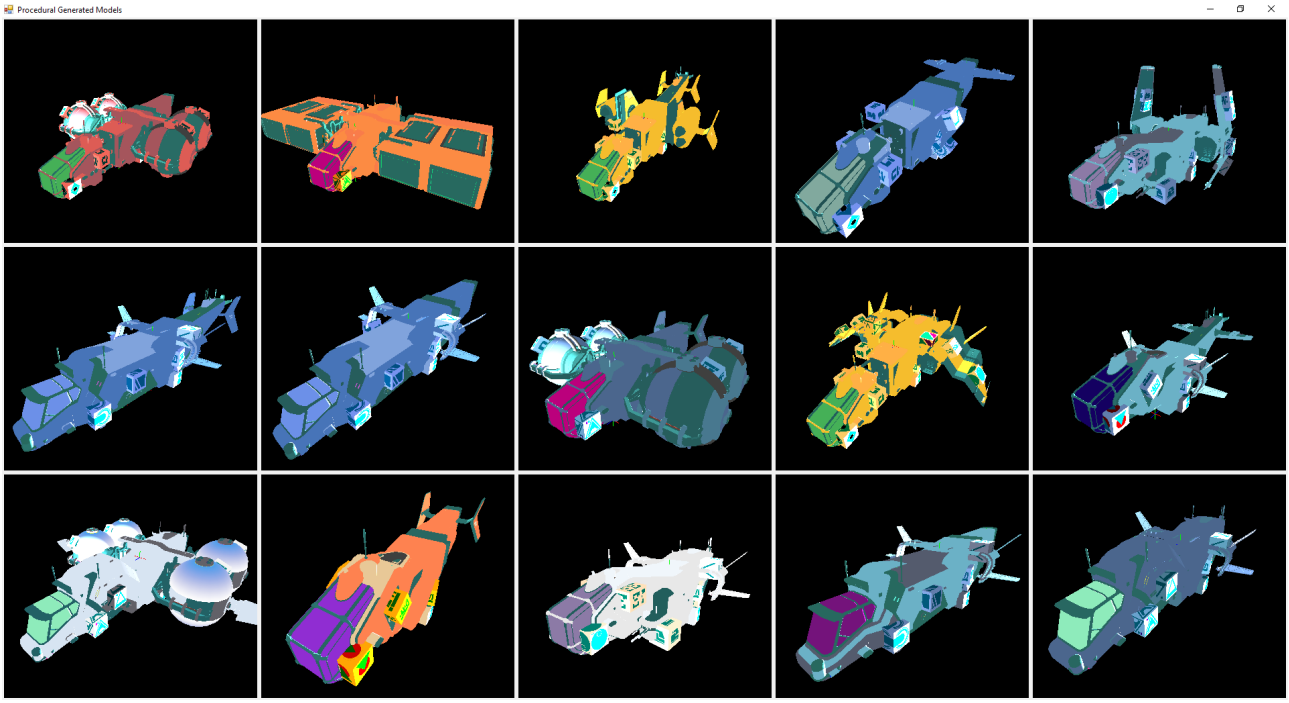
Ключом к этому процессу является то, каким образом сделана селекция частей. Если ближе присмотреться к xml-коду сверху, то можно понять, что каждая часть имеет свойство “Chance” (вероятность), однако его значение стоит на «0» в большинстве случаев. Я полагаю, что настоящие вероятности выбора частей устанавливаются в процессе запуска движка, или они определяются другими файлами-параметрами. В моей программе «model viewer» я рандомизировал эту селекцию: сделал равные шансы для выбора всех частей; и

это привело к созданию крайне странных моделей.

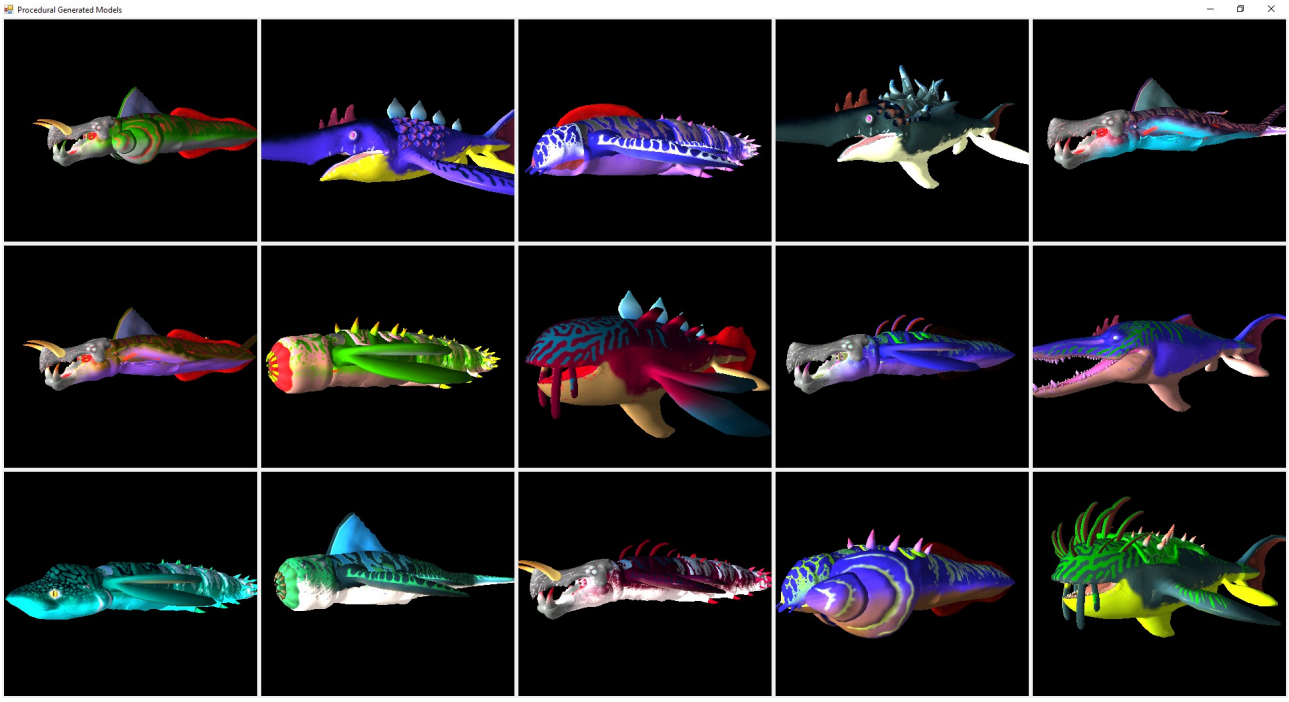
Рандомно созданное существо на основе модели трицератопса



Боевой космический корабль



Грузовой корабль



SHARKRIG

Модель Диплодока из E3-трейлера.

Лично я провел в игре порядка 70 часов, и все это время я НИКОГДА не встречал существа хотя бы похожих на диплодока. Это значит, что движок ошибается, и эти части не избираются (в чем я сомневаюсь), или же их шанс появления настолько мал, что они очень редки в игре. Много дискуссий (чаще всего негативных) прошло из-за контента, которые есть в игре, и того, что был показан в геймплейных трейлерах и подобных вещах. Я не могу говорить об базисной функциональности игры и ее геймплейных особенностях и пр., однако, тщательно проанализировав все модели существ в игровых файлах, я могу сказать, что есть ТОННЫ контента, который по причине решений движка почти всегда (или всегда) не появляется в игре. Если вы спросите меня, то я отвечу, что процедурно сгенерированные модели диплодоков в десять раз лучше, чем статические, и все-таки, если бы они хотели, они мог ли бы легко заставить свой движок загружать статические модели (и, конечно же, весь этот контент из трейлера). Хорошо это или плохо, но это скорее всего дизайнерское решение.

И вот таким образом основная часть процедурно генерируемых элементов работает. Это очень изящный и умный процесс, потому что художники могут очень легко добавить новый контент для процедурной генерации. В действительности, для каждой новой части, которую они добавляют, экспоненциально увеличивается новый номер всех комбинаций (если эта часть будет доступна во всех ветвях дерева). Насколько мне известно, у них было два или три художника, работающих над моделями. Но поразительная вещь в процедурной генерации заключается в том, что, если бы у них было бы в два раза больше людей, работающих ИСКЛЮЧИТЕЛЬНО над этой частью, то игровой контент (только для существ) был бы в сотни раз больше. И даже только этот факт показывает мне возможности и потенциал того, на что движок

NMS способен.

ТЕКСТУРЫ

Текстуры – это еще один очень сложный вопрос в NMS моделях. Как я уже заметил в разделе «Геометрия», все части полигональной сетки определяются SCENE.MBIN-файлами. Записи в этих файлах выглядят таким образом:

```
<Property value="TkSceneNodeData.xml">
    <Property name="Name" value="_Head_Tri" />
    <Property name="Type" value="MESH" />
    <Property name="Transform" value="TkTransformData.xml">
      <Property name="TransX" value="0" />
      <Property name="TransY" value="0" />
      <Property name="TransZ" value="0" />
      <Property name="RotX" value="0" />
      <Property name="RotY" value="0" />
      <Property name="RotZ" value="0" />
      <Property name="ScaleX" value="1" />
      <Property name="ScaleY" value="1" />
      <Property name="ScaleZ" value="1" />
    </Property>
    <Property name="Attributes">
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="BATCHSTART" />
        <Property name="AltID" value="" />
        <Property name="Value" value="19140" />
      </Property>
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="BATCHCOUNT" />
        <Property name="AltID" value="" />
        <Property name="Value" value="5772" />
      </Property>
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="VERTRSTART" />
        <Property name="AltID" value="" />
        <Property name="Value" value="3777" />
      </Property>
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="VERTREND" />
        <Property name="AltID" value="" />
        <Property name="Value" value="4894" />
      </Property>
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="FIRSTSKINMAT" />
        <Property name="AltID" value="" />
        <Property name="Value" value="97" />
      </Property>
      <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="LASTSKINMAT" />
```

```
        <Property name="AltID" value="" />
        <Property name="Value" value="124" />
       </Property>
       <Property value="TkSceneNodeAttributeData.xml">
        <Property name="Name" value="MATERIAL" />
        <Property name="AltID" value="" />
        <Property name="Value"
value="MODELS\PLANETS\CREATURES\TRICERATOPSRIG\TRICERATOPS\HEADTRIMAT
.MATERIAL.MBIN" />
       </Property>
     </Property>
            <Property name="Children">
```

Как вы можете видеть, есть несколько атрибутов, которые я не собираюсь здесь описывать прямо сейчас. Но я расскажу немного о последнем, который определяет, какой материал будет использован в модели. Как видите, этот узел ссылается на материал, который должен быть использован. Если мы мы взглянем нам содержимое файла, то оно выглядит так:

```
<?xml version="1.0" encoding="utf-8"?>
<Data template="TkMaterialData">
  <Property name="Name" value="DiploHeadMat" />
  <Property name="Class" value="Opaque" />
  <Property name="TransparencyLayerID" value="0" />
  <Property name="CastShadow" value="True" />
  <Property name="DisableZTest" value="False" />
  <Property name="Link" value="" />
  <Property name="Shader" value="SHADERS/UBERSHADER.SHADER.BIN" />
  <Property name="Flags">
    ...
  </Property>
  <Property name="Uniforms">
    ...
  </Property>
  <Property name="Samplers">
    <Property value="TkMaterialSampler.xml">
      <Property name="Name" value="gDiffuseMap" />
      <Property name="Map"
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/DIPLOHEAD.DDS" />
      <Property name="IsCube" value="False" />
      <Property name="UseCompression" value="True" />
      <Property name="UseMipMaps" value="True" />
      <Property name="IsSRGB" value="True" />
      <Property name="MaterialAlternativeId" value="" />
      <Property name="TextureAddressMode" value="Wrap" />
      <Property name="TextureFilterMode" value="Trilinear" />
      <Property name="Anisotropy" value="0" />
    </Property>
    <Property value="TkMaterialSampler.xml">
      <Property name="Name" value="gMasksMap" />
      <Property name="Map"
```

```
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/DIPLOHEAD.MASKS.DDS"
/>
     <Property name="IsCube" value="False" />
     <Property name="UseCompression" value="True" />
     <Property name="UseMipMaps" value="True" />
     <Property name="IsSRGB" value="False" />
     <Property name="MaterialAlternativeId" value="" />
     <Property name="TextureAddressMode" value="Wrap" />
     <Property name="TextureFilterMode" value="Trilinear" />
     <Property name="Anisotropy" value="0" />
    </Property>
    <Property value="TkMaterialSampler.xml">
     <Property name="Name" value="gNormalMap" />
     <Property name="Map"
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/DIPLOHEAD.BASE.NORM
AL.DDS" />
     <Property name="IsCube" value="False" />
     <Property name="UseCompression" value="True" />
     <Property name="UseMipMaps" value="True" />
     <Property name="IsSRGB" value="False" />
     <Property name="MaterialAlternativeId" value="" />
     <Property name="TextureAddressMode" value="Wrap" />
     <Property name="TextureFilterMode" value="Trilinear" />
     <Property name="Anisotropy" value="0" />
    </Property>
  </Property>
</Data>
```


Важная часть в содержимом файлов – раздел «Samplers» (Образцы). Очевидно, что эта секция определяет текстуры, которые должны быть использованы на части модели. Вот это интересный момент. На статических моделях спокойно используются заранее созданные качественные текстуры. НО когда полигональная сетка используется в процедурно генерируемой модели, только нормальная текстура является подходящей. Соединяющая текстура, содержащая всю цветовую информацию части, – это пустая белая текстура.

Сначала я думал, что цвета и текстуры устанавливаются только в момент работы движка, но не в этом случае. Вместе с этими текстурными файлами всегда идут “.TEXTURE.MBIN”-файлы, которые, как я считаю, ведут себя точно так же, как модели в файлах-дескрипторах. Они определяют способ, с помощью которого комбинируются текстуры для составления соединения текстур в финальной модели. Дизайнеры игры, кроме тех, кто занимался созданием различных частей моделей, изобрели множество различных текстур для каждой части. Так, прочитывая этот файл так же, как это делает файл-дескриптор, можно высчитать конечное соединение текстуры процедурно генерируемой модели. И здесь еще интереснее. Даже если две модели являются геометрически идентичными, благодаря процедурной генерации текстур, в финальной версии они могут иметь совершенно отличные цвета, отметки, тени и т.д.

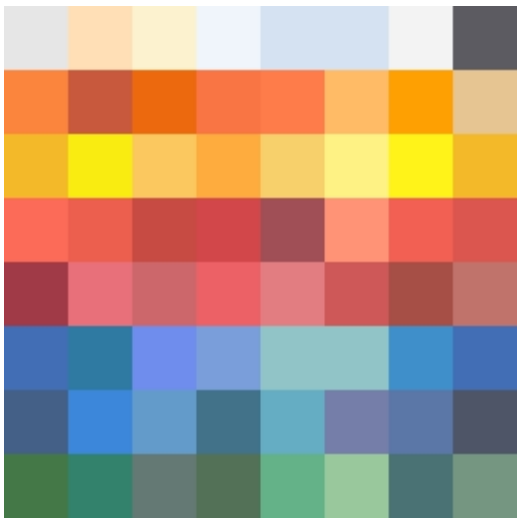
Образец текстур-отметок

В ходе процедурной генерации детали текстур могут значительно отличаться. Выглядит так, будто бы текстуры выстраиваются, наслаиваясь друг на друга. Я приведу пример процедурно сгенерированной текстуры существа. Обычно, в нижнем слое есть основная текста, которые накладывают основной цвет и тень на модель (имена таких текстур заканчиваются на .BASE.DDS). На следующем слое есть текстура нижней части живота (.UNDERBELLY.DDS), которая делает более детальной часть живота существа. Затем следует другой слой, добавляющий больше деталей на рандомные части модели (.UNDERLAYER.X.DDS). Потом, идут отметки (.MARKINGS.X.DDS), определяющие наиболее заметные детали поверхности модели. Следующий слой снова добавляет немного деталей на поверхности отметок модели (.SKIN.DDS). Наконец, на последнем слое есть другая текстура (TOP.X.DDS), которая сильнее детализирует специфические части модели.

Кажется, что это максимум возможных слоев, использованных в процедурной генерации текстур, т. е. восемь (обычно, пять или шесть). Очевидно, что многие текстуры созданы для того, чтобы смешиваться с другими. Именно поэтому все текстуры сопровождаются соответственными «пустыми» текстурами, содержащими альфа-информацию, чтобы смешивание происходило настолько чисто, насколько это возможно. Также основную часть времени вместе с текстурами идет соответствующая карта, регулирующая детали каждой части.

Если этой беспорядочной текстуры недостаточно, то даже с помощью смешения, финальная текстура не будет чисто окрашена. Это еще один гениальный трюк от разработчиков. То, чего они хотели достичь, заключается в том, что каждое существо должно иметь схожую окраску и то, как это происходит в игре с использованием палитры. Я не уверен на 100% относительно того, как это работает в игре, но я собираюсь описать это, и это должно быть весьма точным (потому что я использовал это в своей программе – Model Viewer).

FURPALETTE.DDS



PAINTPALETTE.DDS

Итак, в процессе создания планеты (или системы) отбираются различные цвета, который будут использоваться для ее популяции. Я говорю именно о выборе цвета, потому что в игровых файлах есть весьма необычные цветовые палитры размером 8x8 (PCBANKS/TEXTURES/PALETTES). Этим палитрам соответствуют различные оттенки. Это значит, что 64 цвета, существующих в палитре обыкновенно группируются по четыре или восемь цветов. Таким образом, когда игра прогружает окружающую среду планеты, происходит выбор между группами, которые будут использованы позже. Эти группы легко определить, когда вы смотрите на палитры, потому что они состоят из градиентов между двумя пограничными цветами.

Индексирование внутри отобранной группы производится с помощью информации в .texture.mbin-файле. Записи текстуры в этих файлах выглядят так:

```
<Property value="TkProceduralTexture.xml">
    <Property name="Name" value="GRADIENT" />
```

```
      <Property name="Palette" value="TkPaletteTexture.xml">
        <Property name="Palette" value="Fur" />
        <Property name="ColourAlt" value="Alternative1" />
      </Property>
      <Property name="Probability" value="1" />
      <Property name="TextureGameplayUse" value="IgnoreName" />
      <Property name="OverrideAverageColour" value="False" />
      <Property name="AverageColour" value="Colour.xml">
        <Property name="R" value="0" />
        <Property name="G" value="0" />
        <Property name="B" value="0" />
        <Property name="A" value="0" />
      </Property>
      <Property name="Diffuse"
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/ALIENDIPLO.MARKINGS.
GRADIENT.DDS" />
      <Property name="Normal"
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/ALIENDIPLO.MARKINGS.
GRADIENT.NORMAL.DDS" />
      <Property name="Mask"
value="TEXTURES/PLANETS/CREATURES/TRICERATOPSRIG/ALIENDIPLO.MARKINGS.
GRADIENT.MASKS.DDS" />
            </Property>
```

И снова я не буду разбирать, чем занимаются остальные опции. Что конкретно нас интересует в записях, так это свойство “Palette” (палитра). Учитывая эту информацию, мы можем понять, какой цвет выбирается для нашей части. В этом случае можно видеть, что мы должны индексировать “Fur” (мех, шкура) палитры (которые идут в группах по четыре), и в отобранной группе нам нужен цвет – “Alternative1” (альтернативный). Так индексируются цветовые палитры. Значения свойств “ColourAlt” могут быть такого рода: “Primary” (первичный), “Alternative1”, “Alternative2″,”Alternative3” и пр. Это означает, что “Primary” будет первым цветом группы, “Alternative1” – вторым и т. д. Я не абсолютно уверен, но я так это использую, и в этом есть смысл по причине природы цветовых групп в палитрах.

Теперь у нас есть один базовый цвет для особой текстуры, и что нам, собственно, с этим делать? Все просто – размножить его с помощью цвета текстуры. Стандартно игровые текстуры нейтрального синеватого цвета. После размножения с помощью палитры текстура приобретает более яркий цвет.

Это делается для каждой текстуры, применяемой к модели. Если их восемь, то мы просто извлекаем их, комбинируем с чистыми цветами палитры, затем смешиваем снизу вверх и накладываем на модель.

С помощью всех этих комбинаций объединяющие текстуры могут ПО-НАСТОЯЩЕМУ варьироваться и придают совершенно той же геометрической модели совершенно иную внешность. Техника процедурного генерирования текстур, совмещенная с процедурной генерацией модели, способна создавать необычные модели с уникальным внешним видом. Как я писал выше оо процедурной генерации модели, если бы специальные художники хорошо

проработали модели и добавили бы больше вариаций слоев и более детальные палитры цветов, результат был бы гораздо богаче.

АНИМАЦИИ

Честно говоря, я не изучал анимации так же глубоко, как и геометрию с текстурами. Все, что я сделал (это было не так легко, как может прозвучать), так это парсировал и небезуспешно «проиграл» их в Model Viewer. В этой категории есть много интересного и необычного.

Во-первых, скелет модели определяется “SCENE.MBIN”-файлами. Они представляют собой иерархические узловые системы, к которым прикрепляются части конкретной модели с помощью скелетной анимации (vertex skinning). Здесь ничего необычного. Как я уже упоминал в секции о геометрии, самое интересное тут – множественные части полигональный сетки, находящиеся в SCENE.MBIN файлах. Чтобы урегулировать анимацию и движение всех частей, узловая система определяет ВСЕ части сцены.

Анимации работают точно таким же образом, они анимируют весь скелет, даже если наличие узлов используется для финальной сгенерированной модели. На первый взгляд это выглядит как пустая трата ресурсов. В действительности же только верхняя часть имеет парсирование файла полной анимации. Во время запуска SCENE и GEOMETRY MBIN-файлы производят существенную информацию, чтобы загрузить необходимые данные для каждой части модели в графический процессор.

Играя в игру, я знаю, что есть финальный процедурный процесс, который связан с общим скелетом модели. В некоторых случаях движок может модифицировать этот скелет: понизить центр тяжести модели, сделать ноги короче/длиннее, изменить размер головы и прочее. Я еще не изучил достаточно этот вопрос, чтобы понять, как это контролируется, но мне известно, что это происходит. В целом, это добавляет совершенно новое измерение процедурной генерации существа, потому что так можно изменять окончательный размер модели. По сути, финальный размер можно модифицировать настолько сильно, что существо почти не будет похоже на его исходную модель. Также требуется вычисление инверсной кинематики (Inverse kinematics) для наложения старых анимаций на новый скелет и получения измененных анимаций, которые зависят от сложности модификации и исключительно преображают существа в игре.

ЗАКЛЮЧЕНИЕ

Я попытался здесь объяснить, как работает эта игра, после того, как изучал ее в течение трех недель. Я сконцентрировался на процедурной генерации существ, но те же принципы работают и в остальных случаях (космические корабли, NPC, строение, растения). Когда я начинал исследование файлов, все восхищались игрой, все выглядело таким новым, особенным, но спустя несколько недель все стало одинаковым. Итак, вопрос в том: стоит ли процедурная генерация того, что ею достигается в NMS?

Тут нет простого ответа на вопрос.

Как разработчик ПО и реверсивный инженер, я сказал бы, это АБСОЛЮТНО стоит того. Технически я никогда не видел подобных механик в видеоиграх, кроме как в NMS. И я сомневаюсь, что увижу это в ближайшее время в других играх, которые будут использовать процедурную генерацию, потому что никто не рискнет создавать фэнтезийный мир с достаточным количеством рандомности. С точки зрения технической перспективы NMS является настоящей ЖЕМЧУЖИНОЙ, и все, кто попытается это отрицать, будут лгать сами себе, или они просто не имеют представления (или им все равно) о том, как работают игры. Этот игровой движок имеет очень большой потенциал, и я просто не могу перестать думать о том, какую игру мы могли бы иметь, если бы он был в руках гораздо более крупной студии. Даже с условиями ограниченных изначальных ассетов я остаюсь весьма довольным вариациями существ, которых мы не встречаем в игре (Хотя я очень хочу верить, что с помощью обычной настройки мы можем получить даже больше из существующих ассетов).

Как у игрока, мои чувства довольно смешанные. Даже если это не мой основной стиль игры, я знал еще с момента, когда я предзаказал NMS, что это игра, где я могу просто отдохнуть. Изучать окружающую среду, растения, существ. На первый взгляд может показаться, что вы уже их видели, но если приглядеться – большинство из них будут другими. Вероятно, что текстуры будут отличаться или маленький рог на голове какого-либо существа, или его отметки, или какая-нибудь небольшая деталь космического корабля. Контент там ЕСТЬ (даже геймплейный, и разработчики могли создать его больше, если бы захотели), вы не можете сказать, что его нет. И в итоге вы получаете очень хороший процесс процедурной генерации. Контент, создаваемый движком для одной системы с двумя-тремя планетами, КАК МИНИМУМ превышает количество ассетов, которые вы можете увидеть, например, в видеоигре «ARK». И конечно же, ничто не будет выглядеть идеально на 100%, как все динозавры в ARK, но в этом-то и прелесть. Данный движок способен на производство самых великих и прекрасных существ, и в то же время самых ущербных, которые когда-либо бывали в видеоиграх. Вот почему я купил эту игру, и почему я люблю ее. Я живу для того момента, когда после долгого и нудного изучения, я внезапно приземлюсь на самую красивую планету. Я не сравниваю RPG-элемент или геймплейные особенности. Я говорю только о процедурно генерируемом контенте. Это не значит, что не может быть ничего лучше. Я считаю, что может, и я жду, когда они это сделают.

С другой стороны, если кто-нибудь не хочет расслабляться, быть терпеливым и уделять внимание деталям, то это абсолютно не стоит того, и весь процедурно генерируемый контент является тратой ресурсов. Это не экшен-FPS игра, где ты должен все время охотиться и бежать вперед каждую секунду, и если разработчики что-то и имели в виду, то именно это. Все будет выглядеть абсолютно одинаковым, если кто-нибудь будет играть таким образом. Даже если бы в игре был контент как в трейлере, и на каждой второй планете были и растительность, и огромные динозавры, и песчаные бури, то она все равно была бы скучной, т. к. то же самое будет в третей системе из-за того, что вы просто не

присматриваетесь к этому контенту. Вы не можете обвинять игру или разработчиков, не исследовав ничего в игре об исследовании мира. Кроме того, процедурная генерация не является средством создания контента из ничего так, чтобы он выглядел логичным. По крайней мере, так будет еще в ближайшем будущем. Вы не можете просто засунуть в игру математическую формулу и создавать существ с ее помощью. Вы можете делать это в отношении ландшафта, растений, скал и космических кораблей, но не в отношении живущих движущихся существ и NPC, это настолько сложно, что скорее всего невозможно (и конечно же, если это было бы возможно, то это бы уже использовалось, т. к. нет никаких причин не делать этого). Поэтому если кто-то ожидает увидеть по-настоящему различных между собой инопланетян на каждом шагу, то это проблема желающего, а не разработчиков. Если и есть возможность делать такой тип контента, то это только тот способ, который используется в движке NMS.

После всего этого исследования мне понятно, что в игре есть достаточно контента по крайней мере для того, чтобы различать все вещи на каждой планете, поэтому я не могу винить контент или движок за их скупость. Я обвиняю калибровку движка и его конфигурацию, а также проклятый мультиплатформенный релиз игры и издателя. Я на 100000000000000000% уверен, что разработчики просто спешили к релизу игры. И она даже не близка к своему законченному варианту, и даже не достигает 80% своих возможностей игрового движка. Особенно после изучения файлов это КРИСТАЛЬНО ЯСНО. Эта игра ближе к технической демо-версии, чем к полноценной игре. Пытаясь создать один и тот же контент как для ПК, так и для PS4, они просто порубили игру на куски, при этом стремясь к тому, чтобы она нормально запускалась на низких графических параметрах и имела высокий фреймрейт, что убило игру еще больше. Лично я жду МНОЖЕСТВО обновлений. Я могу забыть о большинстве ошибок студии Hello Games, связанных с этим релизом: денежную переоценку игры, недостаточную коммуникацию, даже недостаток особенностей (таких, как мультиплеер, который мне нафиг не сдался), НО что я никогда не прощу, так это тот факт, что, учитывая предрелизную совершенно хаотичную и напряженную ситуацию, они не смогли доделать внутриигровую конфигурацию движка. Чем, собственно, разработчики модификаций к игре сейчас и заняты – копанием в файлах и поиском того, как сделать ТОТ ЖЕ САМЫЙ ДВИЖОК, создать более богатый и различный контент. И в большинстве случаев моддеры преуспевают в этом просто потому, что ВОЗМОЖНО найти более хороший способ работы движка, чем тот, что он демонстрирует сейчас. Все эти опции должны были быть доступны обычному игроку, а не найдены конкретными моддерами. Очевидно, что они решили не делать этого, т. к. хотели, чтобы у игроков была одна и та же вселенная, чтобы они могли делиться своими найденными маршрутами, существами, растениями, что не бессмысленно. Но даже это они не доделали. Собственно, принудили к оффлайн-игре и продемонстрировали, на что способен их движок...

По какой-то причине я убежден, что студия Hello Games рано или поздно доделает свою игру. Нельзя взять и забыть про более, чем четыре года работы

над движком, который по-настоящему великолепен. А тем фанатам теории заговора от Hello Games, я скажу одно: серьезно, ребята, есть тысячи разных способов, с помощью которых разработчики могли бы вытянуть из вас деньги, и это бы случилось гораздо раньше, если бы они действительно этого хотели.

Пусть это игровой движок расцветет!

Грег