ITerrainGraphics. Интерфейс модуля визуализации земли

Abstract

ITerrainGraphics реализуется с использованием библиотек: osg, edCore, renderer + renderer backends. Спроектирован таким образом чтоб рендерить разные территории разными визуализаторами.

В настоящее время ITerrainGraphics частично реализован для:

- Terrain31 (старая земля: Кавказ)
- Terrain41 (новая земля: Невада, Корсика, Азербайджан, Форт Ракер, Персидский залив, ...)

Необходимые понятия:
визуализатор - front-end модуль/модули отвечающие за рендер кадра
выборка - элемент рендеринга для визуализатора, список графических объектов внутри TerrainGraphics.
шейдинг модель - визуальный и технический режим рендера. Простыми словами: какую картинку хочет визуализатор см. Приложение 4. Шейдинг модели. enum enShadingModel.

Использование

ITerrainGraphics инициализируется по ITerrain* с помощью обертки ITerrainGraphicsEntryPoint. Обертка отвечает и за удаление ITerrainGraphics.

Цикл визуализации:
Сначала визуализатором вызвается метод ITerrainGraphics::parse(). В параметрах передается информация о всех выборках требуемых визуализатору на текущем кадре. Для передачи параметров используется интерфейс IParseContext, реализуемый на

стороне визуализатора. Выборки кешируются в TerrainGraphics и валидны до следующего вызова ITerrainGraphics::parse().

Далее для каждой выборки визуализатор вызывает ITerrainGraphics::render(). В параметрах передается id выборки и параметры рендера. Для передачи параметров используется интерфейс IRenderContext, реализуемый на стороне визуализатора.
Для передачи параметров в шейдера (юниформов) используется константный буфер со структурой SharedParams. Текстуры отдаются через метод IRenderContext::sharedTexture(). Подробнее в Рендер выборки.

Вспомогательные методы:
ITerrainGraphics::getBounds() - вернет доступный после parse() бокс элементов попавших в выборку.
ITerrainGraphics::dump() - текстовый дамп графических объектов в выборке.
ITerrainGraphics::forceLoading() - предварительная загрузка ресурсов.

Важно:

- TerrainGraphics никогда сам не устанавливает render target (он же frame buffer). Это задача визуализатора.
- TerrainGraphics не использует render::getRenderTarget() или аналоги для доступа к разделяемым ресурсам. Все такие ресурсы передаются через контекст рендера (IRenderContext)
- origin() при парсинге и рендере должен быть одинаковый

class ITerrainGraphics

Инициализация

Используется обертка ITerrainGraphicsEntryPoint.
usecase:

```
ITerrainGraphicsEntryPoint edtgLoader;
ITerrainGraphics* pTerrainGrapthics =
edtgLoader.createTerrainGraphics(ITerrain* pTerrain, lua_State* options);
...runtime...
edtgLoader.close();
```

ITerrain* pTerrain - см. ITerrain.h
lua_State* options - настройки. см.Приложение 1. Настройки при инициализации

Парсинг сцены

Метод ITerrainGraphics::parse(const render::IParseContext& parseContext).

TerrainGraphics выберет и отсортирует графические объекты требуемые для рендера кадра.
Парамеры:

- Список требуемых выборок. Выборка описывается камерой и списком шейдинг моделей. Шейдинг модели передаются маской.
- модельное время, origin и центр клипмап текстуры.

Все эти параметры передаются через реализацию IParseContext:

`getOptions()`	луа стейт с параметрами парсинга, см Приложение 2. Настройки при парсинге
`time()`	модельное время - используется в TerrainGraphics для процедурной анимации
`origin()`	Начало координат. Определяет пространство в котором задаются позиции. Нужно для борьбы с потерей точности в 32битных float
`clipmapCenter()`	Центр клипмап текстуры. Обычно - позиция основной камеры, но может быть передвинута на любое место (focus of view). Основное требование - не должен быстро перемещаться.
`sampleCount()`	Количество требуемых выборок
`sampleHandle()`	Хендл выборки по индексу, используется в остальных методах работы с выборками.
`sampleCamera()`	Камера выборки. Если несколько выборок используют одну камеру - должен вернуть одинаковый указатель. У камеры есть параметр parentCamera() - его следует использовать для подвыборок.
`sampleShadingModels()`	Список шейдинг моделей (маска), которые будут использованы для рендера выборки. Маска формируется из енума enShadingModel сдвигом: 1<<enShadingModel. Или используется енум enShadingModelMask
`getDump()`	IDump* интерфейс для сброса дампа парсинга. вернет nullptr когда дамп не нужен.

usecase:

```
ParseContext parseContext;

SampleHandle sm[3];
sm[0] = parseContext.addSample(mainCameras[0], SHM_FINALCOLOR);
sm[1] = parseContext.addSample(mainCameras[1], SHM_FINALCOLOR);
sm[2] = parseContext.addSample(mainCameras[2], SHM_FINALCOLOR);
```

```
SampleHandle ss[4];
ss[0] = parseContext.addSample(shadowMapCameras[0], SHM_SHADOWMAP);
ss[1] = parseContext.addSample(shadowMapCameras[1], SHM_SHADOWMAP);
ss[2] = parseContext.addSample(shadowMapCameras[2], SHM_SHADOWMAP);
ss[3] = parseContext.addSample(shadowMapCameras[3], SHM_SHADOWMAP);

SampleHandle spl = parseContext.addSample(pespLightingCamera, SHM_LIGHTMAP);
SampleHandle sh0 = parseContext.addSample(heightMapCamera, SHM_HEIGHTMAP);

terrainGraphics->parse(&parseContext);
```

Рендер выборки

Метод ITerrainGraphics::render(const SampleHandle& sample, const IRenderContext& renderContext).
TerrainGraphics отрендерит графические объекты для указаной выборки.
Параметры рендера передаются через интерфейс IRenderContext.

`shadingModel()`	шейдинг модель. Должна быть из списка шейдинг моделей указаных для этой выборки в parse()
`origin()`	Начало координат. Определяет пространство в котором задаются позиции. Нужно для борьбы с потерей точности в 32битных float
`camera()`	Камера
`prevFrameCamera()`	Камера предыдущего кадра. Необходима для симуляции блюра.
`sharedTexture()`	Общие текстуры.
`sharedStructuredBuffer()`	Общие структурные буфера
`sharedParams()`	константный буфер со структурой SharedParams
`getDump()`	IDump* интерфейс для сброса дампа рендера. вернет nullptr когда дамп не нужен.

Приложение 1. Настройки при инициализации

Параметр lua_State* options в вызове
`ITerrainGraphicsEntryPoint::createTerrainGraphics()`

TODO

Приложение 2. Настройки при парсинге.

Структура lua стейта передаваемого в IParseContext.

TODO

Приложение 3. Разделяемые текстуры и буфера

Текстуры

CloudShadowsMap	Тени от облаков
HeightMap	Текстура высот и типов растительности под камерой
SkyMap	Текстура небы для тумна
LightMap	Текстура освещения поверхности

Приложение 4. Шейдинг модели. enum enShadingModel.

FINALCOLOR	COLOR4	финальный цвет с освещением
DEFERREDSHADING	decomposed	deferred shading - декомпозиция параметров шейдинга
FLATSHADOWS	COLOR4	плоские тени
SHADOWMAP	DEPTH?	shadow map
SHELFMAP	FLOAT2	морское дно
WAVEMAP	FLOAT2	прибой
HEIGHTMAP	FLOAT2	высота поверхности и тип растительности
LIGHTMAP	COLOR4	освещение
MAPTEX	COLOR4	финальный цвет без освещения
MAPALT	COLOR4	
MAP	COLOR4	финальный цвет - политиеская карта
REFLECTIONMAP	COLOR4	отражения
COMPUTE	append buffer	выборка объектов сцены и подготовка буферов для отдачи на рендер
DEFERRED_LIGHTS	COLOR4	применение источников света (deferred shading)
DEPTH_OF_FIELD	COLOR4, COLOR4, COLOR4	дополнительные слои для постпродакшн: маска, motion vector, глубина