




Russian Sound System 2012

Руководство пользователя



Содержание

1. Что это за движок?
2. Возможности
3. Достоинства
4. Справка по командам

4.1. Ядро

- **RSSCreateCore**
- **RSSCore::pause**
- **RSSCore::restore**
- **RSSCore::createListener**
- **RSSCore::getListener**
- **RSSCore::update**
- **RSSCore::release**
- **RSSCore::createSound**
- **RSSCore::createEnvironmentAABB**
- **RSSCore::setDopplerFactor**
- **RSSCore::setDopplerVelocity**
- **RSSCore::setRolloffFactor**
- **RSSCore::getRolloffFactor**
- **RSSCore::getDopplerFactor**
- **RSSCore::getDopplerVelocity**

4.2. Слушатель

- **RSSListener::setUp**
- **RSSListener::setLook**
- **RSSListener::getLook**
- **RSSListener::setUp**
- **RSSListener::setEFXPreset**

- **RSSListener::getLastEAXPreset**
- **RSSListener::getDefaultEffectSlot**
- **RSSListener::update**

4.3. Семплы

- **RSSSample::reserveSample**
- **RSSSample::deleteAllSamples**

1. Что это за движок?

Russian Sound System 2012 - это звуковой движок предназначенный для создания звукового сопровождения в играх. Поддерживает эффекты EAX и EFX. Движок основан на библиотеке OpenAL и сторонних декодерах(OGG Vorbis, и т.п.).

2. Возможности

1. Загрузка следующих форматов - **wav, ogg**
2. Имитация звукового окружения - более 110 заготовок окружения начиная от ангара и заканчивая космическими станциями! Вот полный список:


<ul style="list-style-type: none">• generic• paddedCell• room• bathroom• livingroom• stonerroom• auditorium• concerthall• cave• arena• hangar• carpettedhallway• hallway• stonecorridor• alley• forest• city• mountains• quarry	<ul style="list-style-type: none">• plain• parkinglot• sewerpipe• underwater• drugged• dizzy• psychotic• castlesmallroom• castleshortpassage• castle_medium_room• castle_long_passage• castle_large_room• castle_hall• castle_cupboard• CASTLE_COURTYARD• CASTLE_ALCOVE• FACTORY_ALCOVE• FACTORY_SHORTPASSAGE• FACTORY_MEDIUMROOM
---	---

<ul style="list-style-type: none"> • SPACESTATION_ALCOVE • SPACESTATION_MEDIUMROOM • SPACESTATION_SHORTPASSAGE • SPACESTATION_LONGPASSAGE • SPACESTATION_LARGERROOM • SPACESTATION_HALL • SPACESTATION_CUPBOARD • SPACESTATION_SMALLROOM • SPORT_EMPTYSTADIUM • SPORT_SQUASHCOURT • SPORT_SMALLSWIMMINGPOOL • SPORT_LARGESWIMMINGPOOL • SPORT_GYMNASIUM • SPORT_FULLSTADIUM • SPORT_STADIUMTANNOY • PREFAB_WORKSHOP • PREFAB_SCHOOLROOM • PREFAB_PRACTISEROOM • PREFAB_OUTHOUSE • PREFAB_CARAVAN • DOME_TOMB • PIPE_SMALL • DOME_SAINTPAULS • PIPE_LONGTHIN • PIPE_LARGE • PIPE_RESONANT • OUTDOORS_BACKYARD • OUTDOORS_ROLLINGPLAINS 	<ul style="list-style-type: none"> • FACTORY_LONGPASSAGE • FACTORY_LARGERROOM • FACTORY_HALL • FACTORY_CUPBOARD • FACTORY_COURTYARD • FACTORY_SMALLROOM • ICEPALACE_ALCOVE • ICEPALACE_SHORTPASSAGE • ICEPALACE_MEDIUMROOM • ICEPALACE_LONGPASSAGE • ICEPALACE_LARGERROOM • ICEPALACE_HALL • ICEPALACE_CUPBOARD • ICEPALACE_COURTYARD • ICEPALACE_SMALLROOM • DRIVING_COMMENTATOR • DRIVING_PITGARAGE • DRIVING_INCAR_RACER • DRIVING_INCAR_SPORTS • DRIVING_INCAR_LUXURY • DRIVING_FULLGRANDSTAND • DRIVING_EMPTYGRANDSTAND • DRIVING_TUNNEL • WOODEN_ALCOVE • WOODEN_SHORTPASSAGE • WOODEN_MEDIUMROOM • WOODEN_LONGPASSAGE • WOODEN_LARGERROOM
---	--

<ul style="list-style-type: none"> • OUTDOORS_DEEPCANYON • OUTDOORS_CREEK • OUTDOORS_VALLEY • CITY_STREETS • CITY_SUBWAY • CITY_MUSEUM • CITY_LIBRARY • CITY_UNDERPASS • CITY_ABANDONED • SMALLWATERROOM 	<ul style="list-style-type: none"> • WOODEN_HALL • WOODEN_CUPBOARD • WOODEN_SMALLROOM • WOODEN_COURTYARD • MOOD_HEAVEN • MOOD_HELL • MOOD_MEMORY • DUSTYROOM • CHAPEL
--	--

3. Специальный объект EnvironmentAABB(Environment Axis-Aligned Bouding Box) - позволит Вам создавать потрясающе реалистичное звуковое окружение в разных частях вашего игрового мира! EnvironmentAABB представляет собой область с настроенными параметрами окружения, при входе в которую автоматически сменяются настройки окружения. Иными словами - зашли в ангар и выстрельнули там из дробовика - будет очень сильное оглушающее эхо несмотря на то, что в исходном звуковом файле нет даже намека на эхо. Впрочем это нужно услышать самому!

3. Достоинства

- 
1. Движок прост в освоении - инициализация занимает всего одну строчку кода!
По всем командам есть исчерпывающая информация. Простота подчеркивается еще и тем что, чтобы в Вашем приложении заиграла Ваша любимая музыка нужно всего лишь четыре(!) строчки кода!
 2. В движке присутствует механизм кэширования сэмплов - иными словами - Вам не нужно следить за тем "А был ли уже загружен этот файл?": если был загружен - Вам будет предоставлен уже загруженный вариант.
 - 3.

4. Справка по командам.

Примечание: все возвращаемые указатели содержатся во внутренних списках движка. Таким образом Вам не нужно заботиться об утечках памяти - при выходе из приложения вызовите `RSSCore::release()`, чтобы освободить все занятые ресурсы. **Удаление(`delete ptr`) указателя, полученного командами движка, приведет к непредсказуемым последствиям!**

Команды ядра.

- **`RSSCore * RSSCreateCore()`** - функция создает ядро и возвращает указатель. Никакая функция из библиотеки не может быть использована до инициализации ядра.
- **`void RSSCore::pause()`** - ставит на паузу все проигрываемые на текущий момент звуки
- **`void RSSCore::restore()`** - продолжает проигрывание звуков

- **RSSListener * RSSCore::createListener()** - создает нового слушателя и возвращает на указатель на него.
- **RSSListener * RSSCore::getListener()** - возвращает указатель на уже созданного слушателя.
- **void RSSCore::update()** - функция заменяет собой последовательность вызовов:
 - void RSSListener::update()**
 - void RSSSound3D::updateAllSounds()**
 - void RSSEnvironmentAABB::updateAll()**
- **void RSSCore::release()** - освобождает все ресурсы - выгружает все семплы, удаляет все звуки - удаляет все. Все полученные указатели становятся не действительными.
- **RSSSound3D * RSSCore::createSound(RSSSample * sample)** - создает новый звук из указанного сэмпла и возвращает указатель на созданный звук.
- **RSSEnvironmentAABB * RSSCore::createEnvironmentAABB(const char * presetName)** - создает новое окружение с параметрами указанными в заготовке с именем presetName. Возвращает указатель на созданное окружение. Названия заготовок приведены выше.
- **void RSSCore::setDopplerFactor(float factor)** - устанавливает новое значение поправки в формуле для расчета частоты
- **void RSSCore::setDopplerVelocity(float vel)** - устанавливает скорость в формуле эффекта Доплера

- **void RSSCore::setRolloffFactor(float rolf)** - устанавливает значение эффекта удаления. Т.е. на сколько будет уменьшаться громкость звука с увеличением расстояния от звука до слушателя
- **float RSSCore::getRolloffFactor()**- возвращает значение эффекта удаления
- **float RSSCore::getDopplerFactor()**- возвращает значение поправки эффекта Доплера
- **float RSSCore::getDopplerVelocity()** - возвращает скорость в формуле эффекта Доплера

Слушатель

- **void RSSListener::setUp(vec3 u)** - устанавливает вектор указывающий поворот слушателя относительно оси OY по умолчанию вектор равен **vec3(0, 0, 1)**
- **void RSSListener::setLook(vec3 l)** - указывает вектор взгляда слушателя - по умолчанию вектор равен **vec3(0, 1, 0)**
- **vec3 RSSListener::getLook()** - возвращает вектор взгляда
- **vec3 RSSListener::getUp()** - возвращает вектор поворота относительно оси OY
- **void RSSListener::setEFXpreset(const char * EAXpreset)** - устанавливает общую для всего мира заготовку окружения с именем EAXpreset из общего списка заготовок приведенного выше

- **const char * RSSListener::getLastEAXPreset()** - возвращает имя последней использованной заготовки окружения
- **unsigned int RSSListener::getDefaultEffectSlot()** - зарезервировано
- **void RSSListener::update()** - обновляет позицию слушателя
- **void RSSListener::setPosition(vec3 pos)** - устанавливает позицию слушателя в глобальных координатах
- **void RSSListener::setVelocity(vec3 vel)** - устанавливает скорость движения слушателя
- **void RSSListener::translate(vec3 speed)** - перемещает слушателя в глобальных координатах
- **vec3 RSSListener::getPosition()** - возвращает текущую позицию слушателя в глобальных координатах
- **vec3 RSSListener::getVelocity()** - возвращает текущую скорость слушателя

Семплы

- **RSSSample * RSSSample::reserveSample(const char * fileName)** - загружает семпл из указанного файла и возвращает указатель на него. Семплы автоматически кэшируются.
- **RSSSample::deleteAllSamples** - удаляет все загруженные семплы. Все указатели на созданные семплы становятся не действительными!