Grass

Description

Grass is implemented as masked mesh tiles layered over terrain. Mesh tile is a prebaked set of grass meshes randomly situated on a square area. For each grass type, several random chunks are generated to prevent repetitiveness. One channel vertex texture is used to mask grass to allow for smooth non-square grass layers. For fully visible squares vertex texture fetch is avoided. Another one channel vertex texture is used to offset grass according to terrain height.

Grass tint texture is generated using diffuse textures of levels terrain. Grass is tinted by terrain according to individual grass type settings.

Associated classes and structures

GrassLib	Library that contains grass mesh chunks for various grass types
GrassGen	Class responsible for generation of grass mesh chunks and holding generation settings
GrassChunkGenSettings	Structure that defines chunks of grass meshes generation
GrassMap	Class that holds information on grass placement on level and does grass rendering.

Associated Source Files

GrassLib.h	GrassLib class header
GrassLib.cpp	GrassLib class implementation
GrassGen.h	GrassGen class header
GrassGen.cpp	GrassGen class implementation
GrassMap.h	GrassMap class header
GrassMap.cpp	GrassMap class implementation

class GrassLib

Summary

Contains grass chunk meshes for different grass types.

Important methods

---

bool Save() const

Summary:

Saves grass library. Single library file is used for all levels.

Return value:

true in case of success, **false** otherwise.

---

bool Load()

Summary:

Loads grass library. Single library file is used for all levels. In game mode, in contrast with editor mode, only those grass types which are used on the level are loaded.

Return value:

true in case of success, **false** otherwise.

INT64 GetStamp() const

Summary:

Retrieves date/time stamp of grass library file. This is used to check against GrassGen settings files to identify if grass library file needs to be regenerated.

Return value:

64 bit date/time stamp of grass library file.

---

void Unload()

Summary:

Unloads grass chunks from memory

---

void ReloadTextures()

Summary:

Reload grass textures conforming with new engine texture quality settings.

---

const GrassLibEntry& GetEntry(const r3dString& name) const

Summary:

Get grass entry by name. Failed assertion in case no such entry is found.

Parameters:
name - name of the entry to return.

Return value:

Library entry for *name.*

---

const GrassLibEntry& GetEntry(uint32_t idx) const

Summary:

Return entry with index *idx* in the library. Failed assertion in case idx is out of bounds.

Parameters:
idx - index of the entry to return.

Return value:

Library entry with index *idx* in the library

---

uint32_t GetEntryCount() const

Summary:

Returns total entry count in the library.

Return value:

Total entry count in the library.

---

uint32_t GetEntryIdxByName(const r3dString& name) const

Summary:

Return index of the entry with name *name.* Return uint32_t(-1) if no such entry is found.

Parameters:
name - name of the entry for which to return the index.

Return value:

Index of the entry with name *name.* -1 if no such entry is found.

---

const r3dString& GetEntryNameByIdx(uint32_t idx)

Summary:

Returns name of the entry with index *idx*. Failed assert in case *idx* is out of bounds.

Parameters:

idx - idx of the library entry for which to return it's name.

Return value:

Name of the entry with index *idx*.

---

const Settings& GetSettings() const

Summary:

Return current library settings. '*Settings*' is a structure with the following fields:

float	CellScale	- scale of the grass chunk in world units
float	MaxMeshScale	- maximum allowed size of the grass mesh in world units.
float	UniformMaskThreshold	- grass mask threshold calculated as average per pixel, above which grass chunk is to be considered fully filled (doesn't require to fetch mask vertex texture)
float	BlankMaskThreshold	- grass mask threshold calculated as average per pixel, for which to consider grass chunk to be empty (no need to render)
float	MaxBlankMaskThreshold	- maximum absolute value, above which to consider chunk to be **non-empty.** Calculated for each pixel individually. If at least one pixel is above this threshold, grass chunk is rendered.

Return value:

Settings struct with grass library settings.

---

void UpdateSettings(const Settings& settings)

Summary:

Update grass library with new settings *settings*. May trigger grass chunk regeneration in case new *CellScale* is supplied in *settings*.

Parameters:

settings - new settings to update grass library with.

---

void UpdateEntry(const GrassLibEntry& entry)

Summary:

GrassLibEntry is a structure that holds chunk meshes for specific grass type, and name of that grass type. After calling this function GrassLib either adds new entry to the library, or updates existing one with supplied *entry*.

Parameters:
entry - GrassLibEntry structure with new data. This structure is filled by GrassGen class.

---

void MildUpdateExistingEntry(const GrassLibEntry& entry)

Summary:

GrassLibEntry is a structure that holds chunk meshes for specific grass type, and name of that grass type. After calling this function GrassLib either adds new entry to the library, or updates existing one with supplied *entry*. Assumes updates of all entry members **except** for the following:

IndexBuffer - pointer to grass chunk index buffer
NumVariations - number of chunk variations (with different grass placement to avoid repetitiveness)
VertexBuffs - vertex buffers for different chunk variations

If these members differ in library entry and supplied entry, failed assert takes place.

Parameters:
entry - GrassLibEntry structure with new data.

class GrassGen

Summary

Generates chunks of grass meshes according to config.

Important methods

bool Load()

Summary:

Loads grass config for all available grass types.

Return value:

true if successful, **false** otherwise

---

bool IsNewerThan(INT64 stamp)

Summary:
Checks if any of the grass config file date/time stamps is newer than provided *stamp*.

Parameters:
stamp - date/time stamp value to check against

Return value:
true if at least one config file is newer than *stamp*, **false** otherwise.

---

void Save()

Summary:
Saves all loaded grass type config files.

---

void Save(const r3dString& entryName)

Summary:

Saves grass type config with name *entryName*.

Parameters:
entryName - name of grass type to save.

---

bool GenerateAll()

Summary:

Generates chunks of grass meshes for all loaded grass types and stores them in GrassLib

Return value:

true if successful, **false** otherwise

---

bool Generate(uint32_t idx)

Summary:

Generates chunks of grass meshes for config entry with index *idx*, updates it in GrassLib

Parameters:

idx - index of desired grass type.

Return value:

true if successful, **false** otherwise.

---

void FillTypes(string_vec& oVec)

Summary:

Fills oList with names of grass types. Resulting indexes of strings in the oList correspond to internal grass type config indexes.

Parameters:

oVec - array of strings to fill with grass type names.

---

uint32_t GetTypeCount() const

Summary:

Returns total available grass type count.

Return value:

Number of available grass types.

---

const GrassPatchGenSettings& GetPatchSettings(uint32_t idx) const

Summary:

Returns generations settings for grass type with index *idx*. (See GrassPatchGenSettings).

Parameters:

idx – index of grass type for which to return settings struct.

Return value:

Settings struct for grass type index *idx*.

---

void SetPatchSettings(uint32_t idx, const GrassPatchGenSettings& sts)

Summary:

Sets grass generation settings *sts* for grass type with index *idx*.

Parameters:

idx - index of grass type to set settings for
sts - settings to set.

---

struct GrassChunkGenSettings

Summary

Contains settings for chunks of grass meshes generation.

Field description

Name	Type	Description
Name	r3dString	Name of grass type
Density	float	Master density for grass meshes placement
ChunkSettings	ChunkSettingsVec	Array of chunk of grass meshes generation for each mesh type present in this grass type

struct GrassChunkGenSettings

Field description

Name	Type	Description
MeshName	r3dString	File name of mesh to use with this chunk
TextureName	r3dString	Texture to use with this chunk
NumVariations	int	Number of chunk variations (to hide repetitiveness)
Density	int	Number of grass meshes to duplicate in these chunk
MeshScale	float	Mesh master scale
MinMeshScaling	float	Minimum mesh scaling coefficient (multiplied with *MeshScale)*
MaxMeshScaling	float	Maximum mesh scaling coefficient (multiplied with *MeshScale*)
RotationVariation	float	1.0 to randomly rotate mesh in range of 0..2*PI, 0.0 to avoid random mesh rotation whatsoever.
AlphaRef	float	Alpha reference value for alpha testing
TintStrength	float	Ammount of grass tinting by terrain diffuse texture

class GrassMap

Summary

Stores grass placement on level and renders grass. Splits level area in equal square cells. Each cell corresponds to GrassLib chunk in size.

Important methods

void Init()

Summary:
Inits grass GrassMap class. Syncs GrassMap dimensions with terrain dimmensions.

---

void Close()

Summary:

Frees all allocated resources

---

bool Save(const r3dString& levelHomeDir)

Summary:

Saves grass map using level path *levelHomeDir*

Parameters:

levelHomeDir - path to level for which to save the grass map

Return value:
true if successful, **false** otherwise

---

bool Load(const r3dString& levelHomeDir)

Summary:

Loads grass map using level path *levelHomeDir*

Parameters:
levelHomeDir - path to level from where to load the grass map

Return value:

true if successful, **false** otherwise

---

bool HasGrassCells() const;

Summary:

Returns **true** if has any active grass cells(chunks), **false** otherwise.

Return value:

true if grass map has any active grass cells (chunks), **false** otherwise.

---

void OptimizeMasks()

Summary:

Remove cell masks where all mask values are approximately 1, remove all cells where all mask values are approximately 0.

---

void Paint(float x, float z, float radius, float dir, const r3dString& grassType)

Summary:

Depending on *dir* either paint with *grassType* over level at position { *x, z* } supplied in world units, or erase grass of *grassType*. Use *radius* for brush radius.

Parameters:

x	- x paint position in world units
z	- z paint position in world units
radius	- radius of painting in world units
dir	- -1 if clearing grass, 1 otherwise
grassType	- name of grass type to paint with

---

void UpdateHeight(float x, float z, float radius)

Summary:

Update grass Y placement at point {*x,z*} supplied in world units. Affect only those cells that intersect with circle of given *radius*.

This update may be necessary, because grass chunks are offset in Y direction using separate (from terrain) 1 channel height textures via vertex texture fetch.

Parameters:

x	- x coordinate to calculate update circle from (in world units)
z	- y coordinate to calculated update circle form (in world units)
radius	- radius of the update circle (in world units)

---

void UpdateHeight()

Summary:

Update grass height on all active grass chunks.

This update may be necessary, because grass chunks are offset in Y direction using separate (from terrain) 1 channel height textures via vertex texture fetch.

---

void Draw(const r3dCamera& cam, Path path, bool useDepthEqual)

Summary:

Renders grass using camera *cam,* path type *path,*and depending on *useDepthEqual* either with D3DCMP_EQUAL or D3DCMP_LESSEQUAL value of D3DRS_ZFUNC.

Parameters:

cam	- camera to render grass with
path	- may be one of the following: DEPTH_PATH – fill only depth render target

COLOR_PATH – fill only color render target
COMBINED_PATH – fill depth and color render targets simultaneously
useDepthEqual - whether to use D3DCMP_EQUAL or D3DCMP_LESSEQUAL for depth comparison.

---

void ConformWithNewCellSize()

Summary:

Conform grass cell placement according to new cell(and chunk) size.

---

void ClearCells()

Summary:

Clears all grass cells removing all grass form level.

---

void ClearGrassType(const r3dString& grassType)

Summary:

Clears grass cells of specific grass type.

Parameters:

grassType – name of the grass type to clear.

---

r3dString GetUsedTypes() const

Summary:

Returns a string which enumerates all grass type names used on current levell.

Return value

String containing enumeration of all grass types used on this level.

---

static float GetVisRad()

Summary:

Retrieve effective grass visibility radius (which may be adjusted according to rendering quality settings).

Return value:

Effective grass visibility radius.