Decals

Description

Decals are drawn after opaque deferred pass is complete, and depth render target is filled. Decals are drown as screen aligned quads, with depth render target used to correctly project the decal. Screen aligned quad is optimized to capture only potentially affected area.

Decals support normal & relief mapping.

Decals come in two variations: static decals and dynamic decals. Static decals have no life span, which is indicated by life span of 0. Static decals are saved with the level.

Dynamic decals have non 0 life span and disappear they exceed their life span. Disappearing is done gradually do reduce visual discomfort. Dynamic decals are not saved with level.

Decals are split into several libraries. Each level can be tied to one specific library.

Associated classes and structures

DecalChief	Class responsible for storing decal information and decal rendering.
DecalGameObjectProxy	Class responsible for linking decal to game object for integrating decals with editor manipulators
DecalUndoAdd*	IUndoItem implementation for decals
DecalType	struct that define decal type
DecalParams	struct that defines decal placed on a level

Associated Source Files

DecalChief.h	DecalChief class header
DecalChief.cpp	DecalChief class implementation
DecalProxyObject.h	DecalGameObjectProxy class header
DecalProxyObject.cpp	DecalGameObjectProxy class implementation
DecalChiefUndoRedoActions.h	DecalUndoAdd* classes header
DecalChiefUndoRedoActions.cpp	DecalUndoAdd* classes implementation

class DecalChief

Summary

Implements decals storage, editing and rendering.

Important methods

---

void AddType(const r3dString& typeName)

Summary:

Adds decal empty (default) decal with name *typeName*

Parameters:
typeName - name of the decal type to add.

---

void AddType(const DecalType& type)

Summary:

Adds decal type with params specified in *type.*

Parameters:

type - structure that defines decal type (see DecalType)

---

void RemoveType(const r3dString& typeName)

Summary:

Deletes decal type with name *typeName.*

Parameters:
typeName - decal type name to remove

---

uint32_t GetTypeCount() const

Summary:

Returns decal type count.

Return value:

Decal type count.

---

const DecalType& GetTypeByIdx(uint32_t idx) const

Summary:

Get decal type description by index *idx.* Out of bounds index produces assertion failure.

Parameters:

idx – index of decal type to retrieve.

Return value:

DecalType structure with index *idx*

---

void UpdateType(uint32_t idx, const DecalType& type)

Summary:

Update decal type with index *idx* with new data *type*. All decals currently placed on the level will be updated too.

Parameters:

idx	- index of decal type to update
type	- new data to update with

---

void UpdateAll()

Summary:

Update all decals currently placed on the level according to possible modified decal type data.

---

uint32_t GetStaticDecalCount(uint32_t typeIdx) const

Summary:

Returns count of static decals with decal type index *typeIdx*.

Return value:

Decal count for for decal type width index *typeIdx*.

---

const DecalParams* GetStaticDecal(uint32_t typeIdx, uint32_t idx) const

Summary:

Returns static decal of type with index *typeIdx*, with decal index *idx*. *Idx* can be in range *0.. GetStaticDecalCount(typeIdx) - 1*

Return value:

Static decal of type with index *typeIdx*, with decal index *idx*.

---

void UpdateStaticDecal(const DecalParams& parms, uint32_t idx)

Summary:

Updates decal placed on a level with new decal parameters (position, projection direction etc., see DecalParams). Decal type id is located in *params* struct, hence it is not passed.

Parameters:

parms	- struct that defines decal placement, size etc. Includes decal type index.

Idx - index of static decal of type *parms.TypeID*

---

void RemoveStaticDecal(uint32_t typeIdx, uint32_t idx)

Summary:

Remove static decal with type index *typeIdx*, decal index within that type *idx*.

Parameters:

typeIdx - decal type index
idx - decal index within supplied decal type.

---

void RemoveStaticDecalsOfType(uint32_t typeIdx)

Summary:

Removes all static decals of type with index *typeIdx*

Parameters:

typeIdx - index of decal type to remove all decals for.

---

int PickStaticDecal(const r3dPoint3D& start, const r3dPoint3D& ray, float len) const

Summary:

Casts a ray from point *start* in direction *ray* for a maximum length of *len.* Returns index of the first static decal, bounding box of which intersects with the ray.

Parameters:

start - starting point of the ray casting
ray - direction of the ray casting
len - length of the ray casting

Return value:

Index of the intersecting static decal in common decal pool. Common decal pool includes both static and dynamic decals. -1 in case no decal got intersected.

---

const DecalParams& GetDecal(uint32_t idx) const

Summary:

Selects decal with index *idx* from common decal pool and returns its parameters. Common decal pool includes both static and dynamic decals.

In case *idx* is out of bounds, assertion fail occurs.

Parameters:

idx – index decal in common decal pool.

Return value:
Parameters of decal of decal with index *idx*.

---

int GetDecalIdxInType(int idx) const

Summary:

Returns index of the decal within among other decals of same type. Supplied *idx* is the absolute decal index in common decal pool. Type of this decal is looked up, then an index in the subset of decals of this type is calculated and returned.

Parameters:

idx – absolute index of the decal.

Return value:

Index of the decal with absolute index *idx* in a subsed of decals with the same decal type.

---

const Settings& GetSettings() const

Summary:

Returns common decal settings.

Fields of *Settings* struct are:

float	*MaxDecalSize*	- Maximum decal size. Sizes of other decals are stored as percentage of this size.
float	*ReliefMappingDepth*	- Depth of relief mapping effect on decals, in case it is turned on.
float	*AlphaRef*	- Reference alpha value for alpha rejecting with decals.
r3dString	*LibFileName*	- Decal library name.

Return value:

Settings struct with common decals settings

---

void SetSettings(const Settings& settings)

Summary:

Sets new decals settings. UpdateAll() call is necessary if you need to propagate the changes to all decals.

Parameters:

settings – new decal common settings to apply.

---

int GetTypeID(const r3dString& typeName)

Summary:

Returns index of the decal type with name *typeName*.

Parameters:

typeName - name of the decal type for which to return type index.

Return value:

Index of the decal type which corresponds to name *typeName*.

---

int Add(const DecalParams& params, bool checkProximity)

Summary:

Add new decal to level with parameters *params*. Check for other decals in proximity if *checkProximity* is **true.** If there're decals in proximity and *checkProximity* is **true**, return -1. Otherwise return absolute index of the newly added decal.

Parameters:

params - params for the new decal
checkProximity - whether to check for other decals in proximity and block adding in case they are found.

Return value:

Index of the newly added decal. -1 if decal could not be added (e.g. due to other decals in proximity).

---

void Remove(int idx)

Summary:

Removes decal with index *idx*.

Parameters:

idx - index of the decal to remove.

---

void Move(int idx, const r3dPoint3D& pos, const r3dPoint3D& norm)

Summary:

Move decal with index *idx* to new position *pos* and set new projection direction *norm.*

Parameters:

idx - index of the decal to move.
pos - new position of the decal
norm - new projection direection of the decal

---

bool LoadLib()

Summary:

Loads library with decals types.

Return value:

true if successful**, false** otherwise

---

void SaveLib() const

Summary:

Saves decals library.

---

bool LoadLevel(const r3dString& levelPath)

Summary:

Load decals for level with level path *levelPath.*

Parameters:

levelPath - path of the level to load the decals for.

Return value:

true if successful, **false** otherwise

---

void SaveLevel(const r3dString& levelPath) const

Summary:

Saves decals for level at path *levelPath*.

---

void UnloadTextures(bool staticOnly)

Summary:

Unloads texture that are used by decals. Unloads only textures which are used by static decals in case *staticOnly* is **true**.

Parameters:
staticOnly - wether to unload textures for static decals only.

---

void LoadTextures(bool staticOnly)

Summary:

Loads texture for that are used by decals. Loads only static textures in case *staticOnly* is set to **true**.

Parameters:
staticOnly - wether to load textures for static decals only.

---

void ReloadTextures()

Summary:

Reloads all textures used by decals. Used to make decals textures conform with game's texture quality settings.

---

void Update()

Summary:

Updates decals before rendering. Must be called every frame.

---

void Draw()

Summary:

Renders all visible decals.

---

void DebugDraw()

Summary:

Draw decals' screen aligned quads for debug purposes.

---

void Init()

Summary:

Initializes the decals. Must be called at engine initialization stage.

---

void Close()

Summary:

Frees all resources used by decals. Must be called at engine shutdown stage.

---

void AutoLoadTextures(int idx)

Summary:

Loads texture for decal with index *idx* in case they are not loaded yet. Can happen when decals are edited in the editor. Never happens during the game.

Parameters:

idx - index of the decal, for which to load textures in case they are unloaded

---

class DecalGameObjectProxy

Summary

Links individual decals with editing object gizmos for easier decal editing.

struct DecalType

Summary

Describes decal type.

Structure fields

Name	Type	Description
Name	*r3dString*	Name of the decal type
DiffuseTex	*r3dTexture**	Diffuse texture to use with decals of this type
NormalTex	*r3dTexture**	Normal texture to use with decals of this type. Alpha channel should contain relief mapping offset. Leave white if relief mapping should not affect this decal.
DiffuseTexName	*r3dString*	File name of the diffuse texture
NormalTexName	*r3dString*	File name of the normal map texture
UVStart	*float[2]*	Texture coordinates of decal's upper left corner
UVEnd	*float[2]*	Texture coordinates of decal's bottom right corner
LifeTime	*float*	Decals life time is the time decals remains on the level before disappearing.
ScaleX	*float*	Relative scale of the decal in X direction. The absolute scale is calculated by multiplying *ScaleX* with *MaxDecalSize* from common decal settings.
ScaleY	*float*	Relative scale of the decal in Y direction. The absolute scale is calculated by multiplying *ScaleY* with *MaxDecalSize* from common decal settings.
UniformScale	*int*	Zero if this decal has non-uniform scale. Non-zero value otherwise (*ScaleY* is ignored in this case)
ScaleVar	*float*	Decal scale variation. Randomly affects the scale of newly added decals. A value of 0 forces all decals of this type to be same sized.
RandomRotation	*int*	Zero in case this decal shouldn't be randomly rotated when added to level. Non-zero value otherwise
BackgroundBlendFactor	*float*	Controls how decal is modulated against his background. 0 for decal to remain completely unmodulated, 1.0 for it to be maximally modulated.
ClipFarFactor	*float*	Controls how far decal's projection gets faded out behind the plane of projection
ClipNearFactor	*float*	Controls how far decal's projection gets faded out in frond of the plane of projection.
MinQuality	*int*	Minimum required decoration quality to show this decal. Decoration quality is found in game quality settings.

struct DecalParams

Summary

Describes decal placed on a level.

Structure fields

Name	Type	Description
TypeID	*int*	Index of the decal type in the library
Pos	*r3dPoint3D*	Position of the decal on the level.
Dir	*r3dPoint3D*	Projection direction for the level
LifeTime	*float*	Life time left on this decal.
ZRot	*float*	Rotation of the around along his axis of projection in radians.
ScaleCoef	*float*	Scale coefficient of this decal to allow for random scale variation
Static	*int*	Non-zero if this decal is static, zero otherwise