Terrain

Description

Terrain is split into variable number rectangular tiles. Each tile is a rectangular grid of vertices. It is mapped with 4x8 + 1 layers of diffuse + normal textures. A set of 4 layers constitutes a material. Several materials can be blended within single tile. Triple sided XY, XZ, ZY projection is used for mapping, which allows to avoid texture stretching. Only transitional faces sample all sides, which increases terrain rendering performance. Texture fetching is done at 2 scales, which allows to reduce repetitiveness at far tiles. Final color is modulated by per-vertex color. Terrain supports several lods. Lod morphing is used to make lod switching less apparent. Terrain supports shadow casting. Terrain reduces GPU load at low quality settings by using simpler mapping & and giving up lod morphing.

Terrain height & paint editing is done in engine's editor. A set of different tools as well as heightmap import/export are available.

Associated classes

r3dTerrain	contains terrain rendering & editing code
obj_Terrain	obj_Terrain is used to link r3dTerrain with level saving/loading.

Associated Source Files

Terrain.h	r3dTerrain class header
Terrain.cpp	r3dTerrain class implementation
obj_Terrain.h	obj_Terrain class header.
obj_Terrain.cpp	obj_Terrain class implementation
Terrain_Erosion.cpp	Terrain editing erosion tool implementation.

class r3dTerrain

Summary

Contains terrain rendering & editing code.

Important methods

void SaveData (const char * fileName, bool writeCache);

Summary:

Saves terrain binary data (vertex & index data) for subsequent loading.

Parameters:

fileName	– file name to save r3dTerrain data to.
writeCache	– whether to write index cache for separating XY/XZ/YZ mapping. Writing this cache should be turned off when new level terrain is created & saved.

---

int Load(const char* dir)

Summary:

Load terrain from path *dir*.

Parameters:

dir	- path to load terrain from.

---

void Unload()

Summary:

Unloads terrain data.

---

void UpdatePhysHeightField();

Summary:

Updates PhysX heightfield from internal heightfield for editing – call after modifying terrain's height field with editing tools.

---

void UpdateOrthoDiffuseTexture();

Summary:

Updates orthogonally projected composite diffuse texture for terrain. This texture is used for grass tinting. Call after modifying terrain texturing & color modulation.

---

void ExtractHeightFieldData ();

Summary:

Extracts height field from PhysX terrain objects and stores it into temporary heightfield for editing with terrain tools.

---

template< typename V>
void ExtractColorData();

Summary:

Extracts color data from terrain vertices for editing terrain color with tools. V can be either **HeightNormalVert** or **HeightNormalVertLQ.**

---

void HeightEditingEnter();

Summary:

Enter terrain into height editing mode – should be called before terrain height is to be modified by tools. Calls ExtractHeightFieldData()

---

void HeightEditingLeave();

Summary:
Leaves terrain editing mode. Calls UpdatePhysHeightField()

---

void ReloadTextures();

Summary:

Reloads terrain texture (diffuse & normal layer textures) to conform with texture quality settings.

---

void RecalcLodData();

Summary:

Recalculates lod variables to conform with terrain quality settings.

---

void EditorSplatOn(int matIdx)

Summary:

Turn splat texture for material with index *matIdx* on. In case splat texture is not created yet, it is created.

Parameters:
matIdx – index of material to turn on.

---

void EditorSplatOff(int matIdx)

Summary:

Turn splat texture for material with index *matIdx* off.

Parameters:
matIdx – index of material to turn off.

---

bool IsSplatOn(int matIdx)

Summary:

Returns if material with *matIdx* is turned on or off.

Parameters:
matIdx – index of material to turn check

Return value:
true if material is ON, false otherwise

---

float GetEditHeight(uint32_t x, uint32_t z) const

Summary:

Returns height for terrain point {*x*,*z*}. Terrain point corresponds to terrain vertices – each vertex represent a separate terrain point (vertices on tile edges being exception). WARNING: this functions only if terrain is in edit mode.

Parameters:
x – terrain point *x* for which to return height
y – terrain point *y* for which to return height

Return value:
non-normalized floating point height value

---

float GetHeight(uint32_t x, uint32_t z)

Summary:

Returns height for terrain point {*x*,*z*}. Terrain point corresponds to terrain vertices – each vertex represent a separate terrain point (vertices on tile edges being exception).

Parameters:
x – terrain point x for which to return height
y – terrain point y for which to return height

Return value:
non-normalized floating point height value

---

float GetEditHeight(const r3dPoint3D &pnt)

Summary:
Returns terrain height for world coordinate *pnt.* Y component of *pnt* is ignored – x and z components are used. WARNING: this functions only if terrain is in edit mode – temporary height editing array is used

Parameters:
pnt – world coordinates for which to return terrain height.

Return value:
non-normalized floating point height value

---

float GetHeight(const r3dPoint3D &pnt)

Summary:
Returns terrain height for world coordinate *pnt.* Y component of *pnt* is ignored – x and z components are used.

Parameters:
pnt – world coordinates for which to return terrain height.

Return value:
non-normalized floating point height value

---

void AppendTileCoords(const r3dPoint3D& from, const r3dPoint3D& to, TileCoords* outCoords)

Summary:

Appends tile coordinates to array *outCoords*. Tile coordinates already present in the array are not added. Tiles which intersect with 2D rectangle { *from* , *to* } are added. 2D rectangle is constructed in XZ plane.

Parameters:

from	- x, z are used to define 'upper left' rectangle point
to	- x, z are used to define 'lower right' rectangle point
outCoords	- tile coordinate array to append coordinates to.

---

uint32_t GetEditColor (uint32_t x, uint32_t z) const

Summary:

Get color of terrain point *x*, *z*. Each terrain point corresponds to certain unique vertex (vertices at tile edges being exception). WARNING: works only in terrain edit mode.

Parameters:
x – terrain point *x* for which to return color
y – terrain point *y* for which to return color

Return value:
32 bit (4x8 bit) color value at terrain point { *x*, *z* }

---

bool ImportHeightmap(const char * fileName, float yScale)

Summary:

Import heightmap from DDS image file assuming 0..1 range and set terrain vertical scale. File format is expected to be D3DFMT_R32F. Width and height of the file must match that of terrain heightmap.

Parameters:
fileName – image file name to load heightmap from.
yScale – vertical scale to set for terrain.

Return value:
true if successful, false otherwise.

---

bool ExportHeightmap(const char * fileName)

Summary:

Export normalized heightmap to DDS image file. Heightmap is normalized according to terrain's m_HeightmapSize member variable. D3DFMT_R32F format is used for saving.

Parameters:
fileName - dds file name to save heightmap to.

---

bool UpdateVertexData(const RECT& rc)

Summary:

Update terrain vertices according to edited heightmap array enclosed in rect *rc*.

Parameters:
rc - rect which encloses XZ terrain area to update vertices for.

---

bool UpdateAllVertexData()

Summary:

Update all terrain vertices according to edited heightmap.

---

void Update (bool bShadowMap)

Summary:

Update terrain visibility before rendering

Parameters:

bShadowMap - wether to update terrain visibility for shadow map rendering.

---

void DrawDeferredMultipassInitial()

Summary:

Draw terrain for deferred renderer. Initial pass – for base layer and first material.

---

void DrawDeferredMultipass()

Summary:

Draw terrain for deferred renderer. Subsequent pathes – for second material and up.

---

void DrawMaterialHeavyness()

Summary:

Paint terrain tiles according to number of layers used. Used for debugging purposes. Color changes from green to red as the number of simultaneous layers increases.

---

void DrawDepth()

Summary:

Draw terrain depth in R32F texture. Used to for algorithm which eliminates terrain SSAO artifacts.

---

r3dVector GetNormal(const r3dPoint3D &pnt)

Summary:

Returns terrain normal for world coordinate *pnt.* Y component of *pnt* is ignored – x and z components are used. Terrain normal is calculated using adjecant height values at time of this function call.

Parameters:

pnt – world coordinates for which to return terrain normal.

Return value:
Calculated terrain normal for point *pnt*

---

const MaterialType* GetMaterialType(const r3dPoint3D& pnt) const

Summary:

Returns material type for world coordinate *pnt.* Y component of *pnt* is ignored – x and z components are used. Material type is used to determine footstep sound and decals for given terrain point.

Parameters:

pnt – world coordinates for which to return material type.

Return value:
Pointer to material type for point *pnt*.

---

int GetDecalID(const r3dPoint3D& pnt, uint32_t sourceHash)

Summary:

Returns decal id for world coordinate *pnt* and hash of source weapon name *sourceHash.*

Return value:

Decal id in decal library.

---

float GetTotalWidth() const

Summary:

Returns total terrain width (along X coordinate) in world units.

Return value:

floating point width value in world units.

---

float GetTotalHeight() const

Summary:

Returns total terrain height (along Z coordinate) in world units.

Return value:

floating point height (along Z coordinate) value in world units

---

void ApplyBrush(const r3dPoint3D &pnt, const float strength, const float radius, const float hardness)

Summary:

Applies up/down height editing tool.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used).
strength	- brush strength (influences amount of change per second)
radius	- brush radius
hardness	- regulates brush strength falloff at brush edges

---

void ApplyHeight(const r3dPoint3D &pnt, const float H, const float strength, const float radius, const float hardness)

Summary:

Sets terrain height at level *H* with falloff at brush edges accordng to *hardness.*

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used).
H	- height to set
strength	- brush strength (influences amount of change per second)
radius	- brush radius
hardness	- regulates brush strength falloff at brush edges

---

void ApplySmooth(const r3dPoint3D &pnt, const float radius)

Summary:

Smoothes terrain height.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used)
radius	- brush radius

---

void ApplyErosion(const r3dPoint3D &pnt, const float strength, const float radius, const float hardness)

Summary:

Applies erosion effect at terrain slopes. This brush has no effect on even terrain patches.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used)
strength	- brush strength (influences amount of change per second)
radius	- brush radius
hardness	- regulates brush strength falloff at brush edges

---

void ApplyNoise(const r3dPoint3D &pnt, const float strength, int bPositiveOnly, const float radius, const float hardness)

Summary:

Applies noise effect for terrain height.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used)
strength	- brush strength (influences amount of change per second)
bPositiveOnly	- wether to allow height modification only upwards
radius	- brush radius
hardness	- regulates brush strength falloff at brush edges

---

void ApplyRamp(const r3dPoint3D& rampStart, const r3dPoint3D& rampEnd, const float rampWidthOuter, const float rampWidthInner)

Summary:

Creates ramp from point *rampStart* to point *rampEnd* specified in world units.

Parameters:

rampStart	- point in world coordinates where the ramp starts
rampEnd	- point in world coordinates where the ramp ends
rampWidthOuter	- total width of the ramp in world units
rampWidthInner	- width of the ramp where same elevation is maintained in direction perpendicular to (*rampEnd – rampStart*)

---

void ApplyMaskBrush(const r3dPoint3D &pnt, int opType, int layerIdx, float val, const float radius, const float hardness)

Summary:

Depending on *opType* paints terrain with texture from given layer or erases this layer from terrian.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used)
opType	- 0 if texture is to be erased from terrain, 1 if texture is to be painted on terrain
layerIdx	- layer index in range 0..4*18. Base layer corresponds to 0 index and doesn't support erasing.
val	- amount of texture to add
radius	- brush radius
hardness	- regulates *val* falloff at brush edges

---

void ApplyColor(const r3dPoint3D &pnt, const r3dColor &dwColor, const float strength, const float radius, const float hardness)

Summary:

Applies modulation color for terrain. All terrain layers are modulated.

Parameters:

pnt	- point of application in world coordinates (y is ignored, x and z are used)
dwColor	- color to modulate terrain with
strength	- amount of color to add
radius	- brush radius
hardness	- regulates *strength* falloff at brush edges

class obj_Terrain

Summary

obj_Terrain is derived from *GameObject* and is used to link *r3dTerrain* initialization with level saving/loading.

Important methods