Particles

Description

Particles are rendered in 2 modes. First mode involves rendering into down-scaled render target and thus incurs less pixel load penalty. This down-scaled render target is then overlayed on the final buffer. Second mode renders particles directly into final buffer, generally incurs more pixel cost, but uses early-z in turn.

For both modes, distortion buffer is filled using MRT. This distortion buffer is used for distorting the final buffer during the overlay stage.

For both modes, depth texture constructed during opaque deferred stage is used for soft particles effect.

2 update modes exist for particle systems. The first one does most transforms on CPU and is suited for SM 2.0 hardware. The second utilizes instancing and does most transforms on GPU, reducing CPU<->GPU data flow and offloading CPU significantly. This method is best suited for SM 3.0 hardware and its successors.

Associated Code

`r3dParticleSystem`	Class that defines and instance of the particle system.
`r3dParticleData`	Class that defines particle system “library data”.
`r3dParticleEmitter`	Class that defines particle system’s emitter
`obj_ParticleSystem`	Class that links *`r3dParticleSystem`* game’s level object system.

Associated Source Files

Particle.h	*`class r3dParticleSystem, class r3dParticleData`* definitions
Particle.cpp	*`class r3dParticleSystem, class r3dParticleData`* implementation
Particle_Int.h	Internal particles classes definitions, including *`r3dParticleData`* and *`r3dParticleEmitter`*
obj_ParticleSystem.h	*`obj_ParticleSystem`* definition
obj_ParticleSystem.cpp	*`obj_ParticleSystem`* implementation
ParticleCPU_vs.hls	Particle vertex shader which relies for most particle transformations done on CPU
ParticleGPU_vs.hls	Particle vertex shader which relies for most particle transformation done on GPU with and which utilizes instancing.
ParticleMesh_vs.hls	Vertex shader for particles which use mesh geometry
Particle_ps.hls	Pixel shader for particles
ParticleMesh_ps.hls	Pixel shader for particles which use mesh geometry

class r3dParticleData

Summary

Contains particle ‘library’ data (constant and common for all instances).

This class has the following public fields:

Name	Type	Description
NumInstances	*int*	Counts number of particle system instances that reference this particle system library item.
FileName	*char [256]*	Name of this particle system library item.
TextureFileName	*r3dString*	Diffuse texture file name
NormalTextureFileName	*r3dString*	Normal texture file name
DistortTextureFileName	*r3dString*	Distort texture file name
Texture	*r3dTexture**	Diffuse texture
Atlas	*AtlasDesc*	Atlas that contains texture coordinates for different emitters of the particle system
ZSortEnabled	*int*	Enable(non-zero) or disable (0) depth based sorting of this particle system
WarmUpTime	*float*	Amount of time the particle system gets updated without being show after being created.
DepthBlendValue	*float*	This value regulates sharpness of depth rejection for the particle (particle softness)
OrgGlobalScale	*float*	Global scale of the particle system
EmitTime	*float*	Time the emitters of the particle system are active
EmitterType	*int*	Determines how the way emit positions are chosen in relation particle system position. Can be one of the following values *R3D_EMITTYPE_POINT* Emit position is chosen randomly within the XZ aligned square of *2*EmitRadius* length *R3D_EMITTYPE_LINE* Emit position is randomly chosen along *EmitVector* vector. Length of the vector defines the maximum offset *R3D_EMITTYPE_MESH* Random faces of the mesh *EmitMesh* are used as emitting points. *R3D_EMITTYPE_RING* Emit position is randomly chosen inside a circle of *EmitRadius* radius. *R3D_EMITTYPE_SPHERE* Emit position is chosen randomly inside a sphere of *EmitRadius* radius.
EmitMesh	*r3dMesh**	Faces of this mesh may be randomly chosen as emit points.
EmitVector	*r3dVector*	Point within this vector may be chosen as emit point.
EmitRadius	*float*	Radius, used to determine emit position for various emit position determining modes (*EmitterType)*
bAcceptsDynamicLights	*int*	Determines, whether the particle system can be lit by dynamic lights (point, spot lights etc.)

bCastsShadows	*int*	Determines, whether the particle system casts shadows.
HasLight	*int*	Determines, whether the particle system has attached light.
LightLifetime	*float*	Amount of time the attached light remains active
LightIntensity	*float*	Intensity of the attached light
LightRadius1Base	*float*	Base value of the fade out start radius of the attached light
LightRadius2Base	*float*	Base value of the fade out end radius of the attached light
LightRadius1	*r3dTimeGradient2*	Function with result values in range of 0..1 which are multiplied with *LightRadius1Base* when effective light fade out start radius is calculated
LightRadius2	*r3dTimeGradient2*	Function with result values in range of 0..1 which are multiplied with *LightRadius2Base* when effective light fade out end radius is calculated
LightColor	*r3dTimeGradient2*	Function that defines the color of the attached light
ParticleLightOffset	*r3dPoint3D*	Offset of the attached light with relation to particle system center.
bDeferredMeshParticlesShadows	*int*	In case meshes are used as particles, determines if these meshes will cast shadows.
bDeferredMeshParticles	*int*	Determines whether meshes are used as particles.
bReceivesShadows	*int*	Determines if the particle system can receive shadows
bDistort	*int*	Determines if the particle system has does screen distortion and thus has a distortion texture.
bNormalMap	*int*	Determines if the particle system has normal maps for the particles.
fBumpPower	*float*	Bumpiness parameter for particle's normal mapping mode.
DistortTexture	*r3dTexture**	Distortion texture for *bDistort != 0* case
NormalTexture	*r3dTexture**	Normal texture for *bNormalMap != 0* case
OrgDirection	*r3dPoint3D*	Vector that defines main emit direction of the particle system. All emitters emit in this direction with certain configurable variation.
GlowIntensity	*float*	Intensity of particle's glow effect
GlowFromDistortB_or_FromDiffuseA	*float*	Defines which channel to use as glow source. On option is to use distort texture blue channel (it remains unused otherwise).
PType	*r3dParticleEmitter* [16]*	Emitters to use with this particle system. Inactive emitters are filled with NULL.

class r3dParticleEmitter

Class that describes single emitter within the particle system. It has following public fields:

Name	Type	Description
Name	*char[64]*	Emitter name
bActive	*int*	Determines if the given emitter is active in the

		particle system
StartFrame	int	Starting frame of the texture animation for the emitter among the frames allocated in the texture atlas
FrameRate	float	Frame rate of the texture animation
UVSpeed	float	UV animation speed for mesh particles
StartTime	float	Starting time of emitting
EndTime	float	End time of emitting
ParticleType	int	Type of the particle geometry. May be one of the following values: *R3D_PARTICLE_CAMERAQUAD* *R3D_PARTICLE_ARBITARYQUAD* *R3D_PARTICLE_RAY* *R3D_PARTICLE_MESH* *R3D_PARTICLE_TRAIL* *R3D_PARTICLE_BEAM* See section below this table for detailed description
RayWidth	float	Particle width for *R3D_PARTICLE_RAY, R3D_PARTICLE_TRAIL* and *R3D_PARTICLE_BEAM* particle types
bDirectionOriented	int	For *R3D_PARTICLE_MESH*, determines if the particle should be oriented along its direction of movement.
EmitDistance	float	If non-zero, particle is added each time emitter travels *EmitDistance*
vDirOrientedAdditRotation	*r3dVector*	For *R3D_PARTICLE_MESH* and *bDirectionOriented == true,* defines additional rotation to apply to direction oriented rotation.
bMeshUseDistortTexture	int	For *R3D_PARTICLE_MESH*, if non-zero, defines that individual distortion texture should be used for mesh particles.
Mesh	*r3dMesh**	For *R3D_PARTICLE_MESH*, defines the mesh to be used as particle.
bMeshDisableCulling	int	For *R3D_PARTICLE_MESH*, defines if culling should be disabled (*D3DCULL_NONE*) during rendering
MeshTexture	*r3dTexture**	Diffuse texture to use for *R3D_PARTICLE_MESH* particles.
MeshDistortTexture	*r3dTexture**	Distortion texture to use for *R3D_PARTICLE_MESH* particles
MeshFName	*r3dString*	Mesh file name for *R3D_PARTICLE_MESH* particles
MeshTexFName	*r3dString*	Texture file name for *R3D_PARTICLE_MESH* particles
MeshDistortTexFName	*r3dString*	Distortion texture file name for *R3D_PARTICLE_MESH* particles
bSingleParticle	int	Determines if only single particle is allowed to be emitted.
bRandomDirection	int	If set to non-zero value, random direction is used, scaled by *RandomVector* from this structure. Otherwise general particle system is used,

		deviated according to `DirectionRand` field of this structure.
`bEmmiter`	`int`	Defines is particles, emitted by this emitter are themselves used as emitters.
`bEmitterDistanceSpawn`	`int`	Determines, if particles emitted from sub-emitters (particles of this emitter, used as emitters), emit on distance based basis.
`bUsedAsEmitter`	`int`	Determines if this emitter is used as sub-emitter for some other emitter in this system.
`bEmmiterTypeID`	`int`	Index of emitter within this system, which is used as sub-emitter for this emitter.
`RandomVector`	`r3dVector`	This vector is used as “scale” for random direction for `bRandomDirection!=0` case
`EmmiterOffset`	`r3dVector`	Offset of this emitter from the emitting point determined by `EmitterType` of `r3dParticleData`.
`ParticleLifeTime`	`float`	Base lifetime value for the particles emitted by this emitter.
`ParticleBirthRate`	`int`	Number of particles added per second
`ParticleSize`	`float`	Size of the added particles
`ParticleVelocity`	`float`	Base velocity of the added particles.
`ParticleGravity`	`Float`	Gravity that affects the added particles.
`ParticleSpin`	`Float`	Base rotation of the added particles
`MotionRand`	`r3dVector`	“Scale” of the random offset applied to the particle motion
`MotionRandDelta`	`float`	Delta value, which determines how often new random offset for random particle motion is calculated
`MotionRandSmooth`	`float`	Determines how rigid is the motion between random particle offsets.
`BornPosRand`	`r3dPoint3D`	Scale for one time random offset applied to the particle when it is added.
`DirectionRand`	`r3dVector`	If `bRandomDirection` is turned **off**, this vector defines random deviation from common particle system direction for the newly added particles.
`ParticleLifeTimeVar`	`float`	Variation of `ParticleLifeTime` applied to each particle at birth. Variation is randomly applied either into positive or into negative direction.
`ParticleSizeVar`	`float`	Variation of `ParticleSizeVar` applied to each particle at birth. Variation is randomly applied either into positive or into negative direction.
`ParticleVelocityVar`	`float`	Variation of `ParticleVelocity` applied to each particle at birth. Variation is randomly applied either into positive or into negative direction.
`ParticleGravityVar`	`float`	Variation of `ParticleGravity` applied to each particle at birth. Variation is randomly applied either into positive or into negative direction.
`ParticleSpinVar`	`float`	Variation of `ParticleSpin` applied to each particle at birth. Variation is randomly applied

		either into positive or into negative direction.
TrailStepDist	float	For *R3D_PARTICLE_TRAIL*, determines the length of the trail segment.
TrailSizeCoefMin	float	For *R3D_PARTICLE_TRAIL* . Minimum coefficient value, which is multiplied with base trail width. Random coefficient between minimum and maximum value is chosen for each section.
TrailSizeCoefMax	float	For *R3D_PARTICLE_TRAIL* . Minimum coefficient value, which is multiplied with base trail width. Random coefficient between minimum and maximum value is chosen for each section.
TrailOpacityCoefMin	float	For *R3D_PARTICLE_TRAIL* . Minimum coefficient value, which is multiplied with base trail opacity. Random coefficient between minimum and maximum value is chosen for each section.
TrailOpacityCoefMax	float	For *R3D_PARTICLE_TRAIL* . Maximum coefficient value, which is multiplied with base trail opacity. Random coefficient between minimum and maximum value is chosen for each section.
TrailTaleFade	float	Determines the fading of the trail section opacity towards the tail of the trail.
TrailTaleFadePow	float	Power of trail opacity fading towards the end of the trail.
TrailDrift	float	Amount of random “wind” drifting applied to the trail.
AngleXOverLife	*r3dTimeGradient2*	Chart that defines X particle rotation. X axis of the chart is particle’s life time.
AngleYOverLife	*r3dTimeGradient2*	Chart that defines Y particle rotation. X axis of the chart is particle’s life time.
AngleZOverLife	*r3dTimeGradient2*	Chart that defines Z particle rotation. X axis of the chart is particle’s life time.
ColorOverLife	*r3dTimeGradient2*	Chart that defines particle color. X axis of the chart is particle’s life time.
OpacityOverLife	*r3dTimeGradient2*	Chart that defines particle opacity. X axis of the chart is particle’s life time.
SizeOverLife	*r3dTimeGradient2*	Chart that defines particle size coefficient. X axis of the chart is particle’s life time.
VelocityOverLife	*r3dTimeGradient2*	Chart that defines particle velocity coefficient. X axis of the chart is particle’s life time.
GravityOverLife	*r3dTimeGradient2*	Chart that defines particle gravity coefficient. X axis of the chart is particle’s life time.
SpinOverLife	*r3dTimeGradient2*	Chart that defines particle spin offset. X axis of the chart is particle’s life time.
BindGravityOverLife	*r3dTimeGradient2*	Bind gravity chart. X axis of the chart is particle’s life time. This gravity works in the direction of the movement of the particle system.
BlendModeOverLife	*r3dTimeGradient2*	Blend mode chart. X axis of the chart is particle’s life time. Y value can be set in range of

		[0..1]. 0 corresponds to fully **additive** mode, whereas 1 corresponds to pure **alpha blend** mode.
FramesOverLife	*r3dTimeGradient2*	Charts that regulates frame flow in the particle animation. Y value, that linearly changes from 0 to 1 corresponds to normal frame flow. Deviations from this state either makes playback faster, or slower depending on the values being above, or below the linear [0..1] chart.
BirthSizeOverLife	*r3dTimeGradient2*	Additional chart for size coefficient, which takes particle system life time, scaled by *BirthChartsTimeLapse* as X axis, and applies only to birth particle size.
BirthChartsTimeLapse	*float*	Scaling coefficent of the particle **system** used as input for *BirthSizeOverLife.*

Below are the descriptions of the values *ParticleType* field of r3dParticleEmitter can take

The following conventions are used:

PrevParticlePos – particle position in previous frame.
ViewVector = CameraPos – ParticlePos ;
DirVector = ParticlePos – PrevParticlePos ;

Type	Description
R3D_PARTICLE_CAMERAQUAD	Standard camera aligned particle.
R3D_PARTICLE_ARBITARYQUAD	Each particle is aligned according to its x, y, z rotation angles
R3D_PARTICLE_RAY	*DirVector* defines X axis. Cross product of *DirVector* and *ViewVector* defines the Y axis.
R3D_PARTICLE_MESH	Custom free oriented mesh geometry is used for each particle. One source mesh is used for all particles of the emitter.
R3D_PARTICLE_TRAIL	As the particle moves, its positions are saved each time the particle passes configurable constant distance. A quad is built between each 2 consecutive positions. For each 2 positions *DirVector* is calculated. This *DirVector* defines the X axis. Cross product of *DirVector* and *ViewVector* calculated for one of the 2 positions serves as Y axis.
R3D_PARTICLE_BEAM	A beam is drawn from the particle system position, and till the *BeamTargetPosition* of *r3dParticleSystem* class. Direction from particle and to *BeamTargetPosition* is used as particles X axis. Cross product of this direction and *ViewVector* is used as Y axis.

class r3dParticleData

Description

This class represents an instance of the particle system.

It has the following important fields.

Name	Type	Description
PD	*const r3dParticleData**	Library item which describes particle system properties common for all instances.
ParticleLight	*r3dLight*	Light, attached to this particle system
NumTrisToDraw	*int*	Number of triangles to draw. Counted dynamically as particles are created and destroy.
PrevPosition	*r3dPoint3D*	Position of the particle system in the previous update frame
Position	*r3dPoint3D*	Position of the particle system
Direction	*r3dVector*	General direction of the particle system. May be ignored by certain emmiters.
DeflectorTop	*float*	Defines XZ plane above which particles may not go, and which deflects incoming particles.
DeflectorBottom	*float*	Defines XZ plane below which particles may not go, and which deflects incoming particles.
BBox	*r3dBoundBox*	Dynamically updated bounding box of the particle system.
MaxBBoxSize	*float*	Maximum bounding box size, traced for particle system LOD calculation.
SourceMoveDelta	*r3dPoint3D*	Difference between *Position* and *PrevPosition*
IsVisible	*bool*	Traces if the particle system is visible.
StartTime	*float*	Time when the particle system was started.
PrevTime	*float*	Previous update time.
TimePassed	*float*	Time, passed since *StartTime*
bEmit	*int*	This variable controls if particle system is allowed to emit particles.
bRenderUntextured	*int*	Debug control variable which indicates if textures are allowed to be used during this particle's system rendering.
EmitersTypes	*int*	Bit field which enumerates all emitter types (see *ParticleType* of *r3dParticleEmitter*
BeamTargetPosition	*r3dPoint3D*	Target position for *R3D_PARTICLE_BEAM* emitter types.
GlobalScale	*float*	Controls the scale of the particle system.
Array	*r3dSingleParticle**	Particle array, shared for all emitters in the system.
ArraySize	*int*	Size of the shared particle array
NumAliveParticles	*int*	Count of the alive particles in *Array*
NumAliveQuads	*int*	Separate count for alive quads. This count may differ from *NumAliveParticles* since some particles may use meshes.
LastTimeCreated	*float[16]*	Emitter bound time of the last particle creation for that emitter.
PTypeSpawned	*int[16]*	Holds emitter bound last particle emit count.
ToEmitWithDistance	*float[16]*	Emitter bound number of particles that should be emitted according to particle system travel distance.

Important methods

`void Restart(float CurTime)`

Summary:
Restarts the particle system.

Parameters:

`CurTime`	– game time. Substitute time returned by *`r3dGetTime()`*

`void Update(float curTime, bool bUpdate)`

Summary:

Update the particle system.

Parameters:

`curTime`	– game time
`bUpdate`	– this parameter is used internally and set to **true** by default

---

`void        Draw(const r3dCamera &cam)`

Summary:

Draws the particle system excluding mesh particles. This function is called during transparent rendering stage.

Parameters:

`cam`	– camera to draw with

`void        DrawDefferedMeshes(const r3dCamera &cam, bool bShadowMap)`

Summary:

Draws mesh particles of the particle system. This function is called during deferred buffer filling stage and during shadow casting stage.

Parameters:

`cam`	– camera to use for rendering
`bShadowMap`	– **`true`** if rending is done into shadow map

`void        ClearParticlesOfType( BYTE type )`

Summary:

Removes all particles with emitter index stored in each particle as *type*

Parameters:

`BYTE type`	-	particles of this index are removed