Explosion post process effect

Description

Explosion effect is post process effect that enhances visual perception of exploded entities, such as grenades and mines. Effect is similar to radial blur, with smooth fade-in and fade-out times.

Associated classes and structures

ExplosionVisualController	Manages multiple exposions. Controls fade-in fade-out periods of effect
ExplosionParameters	Explosion parameters structure, that used to define new explosion effect.
PFX_ExplosionBlur	Post process effect configuration class.

class ExplosionVisualController

This class manages multiple explosion effects. Performs world-space to screen-space coordinate conversion, adjusts effects strength depending on viewer distance and schedule post-process effects into post-process chain.

Sources: ExplosionVisualController.h, ExplosionVisualController.cpp

Important methods

void AddExplosion(const r3dVector &pos, float radius, float duration = 0, float maxStrength = 0, float brightThreshold = 0)

Add new explosion effect.

Parameters:
pos – world space position of explosion epicenter
radius – radius of explosion in world units
duration – duration of effect in seconds. If 0 use default duration.
maxStrength – maximum strength of effect. If 0 use default value.
brightThreshold – brighthness threshold value. If 0 use default value.

void SetMaxVisibleDistance(float dist)

Set maximum distance from camera, where post process explosion effect still visible.

Parameters:
dist – maximum visible effect distance in world units.

float GetMaxVisibleDistance()

Get maximum visible distance of effect.

void SetDefaultDuration(float dur)

Set default effect duration. This value is used when AddExplosion function receives zero in duration parameter.

Parameters:
dur – duration of effect in seconds.

float GetDefaultDuration() const

Return default effect duration in seconds.

void SetDefaultMaxStrength(float str)

Set default maximum strength of effects. This value is used when AddExplosion function receives zero in maxStrength parameter.

Parameters:
str – strength of the effect

float GetDefaultMaxStrength() const

Return default maximum effect strength.

void SetDefaultBrightThreshold(float thr)

Set default bright threshold of effects. This value is used when AddExplosion function receives zero in brightThreshold parameter.

Parameters:
thr – brightness threshold

float GetDefaultBrightThreshold() const

Return default bright threshold of explosion effect.

void ApplyPostFXExplosionEffects()

For all active explosions add post-process effects to post-process system. This function act as update loop. When duration of explosion effect is expired, it is removed from internal array.

void RemoveAll()

Forcibly remove all active explosion effects.

struct ExplosionParameters

Structure to store explosion effect creation parameters and state variables.

Sources: ExplosionVisualController.h, ExplosionVisualController.cpp

Important members

float maxStrength
Maximum effect strength.

r3dVector3 pos
Position of effect in world space.

float startTime
Effect start time.

float duration
Effect duration in seconds.

r3dLight light
Attached light object.

float radius
Radius of effect in world units.

float brightThreshold
Brightness threshold of effect.

class PFX_ExplosionBlur

Post process definition class that is used to represent explosion effect. This effect is very similar to radial blur effect. For detailed description please consult post-process effect system documentation.

Sources: PFX_ExplosionBlur.h, PFX_ExplosionBlur.cpp

Usage example

This clas intended to be in single instance, so we create one global manager for all explosions:

```
ExplosionVisualController gExplosionVisualController;
```

Because we need update loop, and ApplyPostFXExplosionEffects function act like update, we should add call in place where all post FX setup reside:

```
void r3dDefferedRenderer::PostProcess()
{
…
  gExplosionVisualController.ApplyPostFXExplosionEffects();
…
}
```

Now we can add explosions to the manager, and it will do all the work related to updating effect state and post fx submit. For example react on explosion packet sent to player:

```
BOOL obj_AI_Player::OnNetReceive(DWORD EventID, const void* packetData, int packetSize)
{
…
  switch(EventID)
```

```
{
…
   case PKT_S2C_SpawnExplosion:
   {
     …
     const PKT_S2C_SpawnExplosion_s& n = *(PKT_S2C_SpawnExplosion_s*)packetData;
     GameObject* from = GameWorld().GetNetworkObject(n.FromID);
     gExplosionVisualController.AddExplosion(from->GetPosition(), n.radius);
     …
    }
…
}
```