LU Computergraphik 2 (186.165)

SS 2009

3. Abgabe

Total Perspective Vortex

Dokumentation

Reinhold Preiner (0430380, 066 932)
reinholdpreiner@gmx.at

Implementierte Effekte

1. Shadow Mapping

Code-Referenzen

Klassen: ShadowMapper, GLDepthTargetView

Shader: DepthMap.psh, ShadowMappedMaterial.vsh, ShadowMappedMaterial.psh

Beschreibung

Unter Zuhilfenahme von OpenGL-FrameBuffer-Objects (FBOs) wird eine Textur als RenderTarget für das Rendern einer Depth-Map aller Shadowcaster-Objekte in der Szene verwendet. Dafür verantwortlich ist die Klasse ShadowMapper, welche ein Register aller Shadowcaster-Geometries verwaltet und in jedem Frame als ersten Pass besagte Depth-Map rendert. Diese DepthMap-Textur wird für den 2. Pass, dem Render der Szene, als Sampler-Variable des Pixelshaders `ShadowMappedMaterial.psh` gesetzt. Dieser Pixelshader berechnet in der Funktion `getPCFLitFactor()` einen Abschattungsfaktor für jedes Pixel, welcher mittels eines 3x3-Kernels einen einfachen Softshadow erzeugt (Momentan kein tatsächliches PCF). Bei totaler Beschattung wird für das Pixel nur mehr der Ambient + Emissive Farbwert verwendet. Bei totaler Beleuchtung wird auch der Diffuse- und Specular-Term hinzuaddiert.

Für die Umrechnung von Vertex-Position auf eine für den Lookup eines Vergleichstiefenwerts in der Shadowmap erforderliche Texturcoordinate ist eine Matrix zuständig, welche ebenfalls die Klasse ShadowMapper verwaltet und dem VertexShader `ShadowMappedMaterial.vsh` als uniform Variable übergibt:

```
uniform mat4 shadowMapTexCoordMatrix;
…
// Calculate the Texcoord for lookup in the ShadowMap texture
shadowMapTexCoord = shadowMapTexCoordMatrix * MMatrix * gl_Vertex;
```

MMatrix stellt die ModelMatrix des aktuellen Objekts dar. shadowMapTexCoordMatrix ist eine Komposition aus ViewprojectionMatrix der Lichtquelle und einer Biasmatrix, welche DeviceCoordinaten aus dem [-1,1]-Bereich auf den [0, 1]-Bereich mappt (siehe `ShadowMapper.cpp`):

```
// Update the Vertex-To-Shadowmap-Texcoord Lookup Matrix
//-----------------------------------------------------------------------
Matrix4F lightSourceVPMatrix = renderer->GetRCViewProjectionMatrix();
m_shadowMapTexCoordMatrix = m_biasMatrix * lightSourceVPMatrix;
```

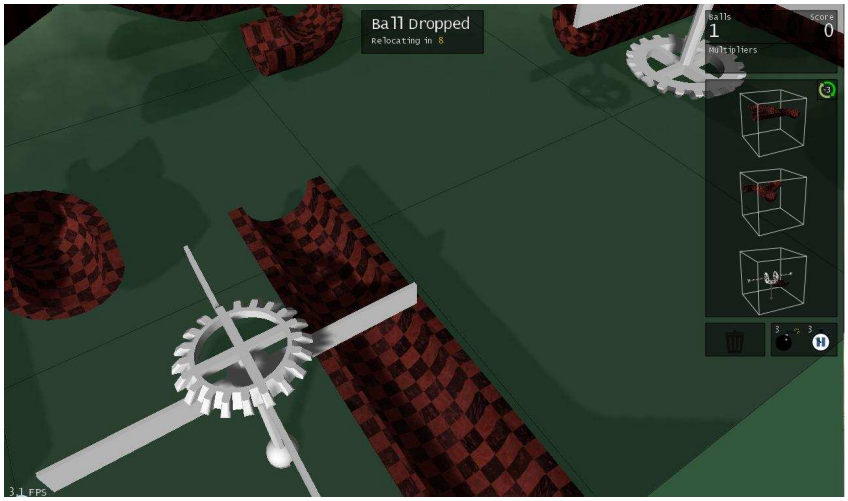


Abb 1.: Szene mit ShadowMapping

2. HIGH-DYNAMIC-RANGE EFFECT MIT BLOOM

Code-Referenzen
Klassen: HDREffect, GLRenderTargetView, GLDepthTargetView
Shader: HDRAdaptedLuminance.psh, HDRBrightPassFilter, HDRDownscaleAvgLuminance4x4.psh, HDRFinalAvgLuminance.psh, HDRFinalSceneCompositing.psh, HDRGaussBlur15.psh, HDRInitialAvgLuminance, HDRDownscale2x2.psh

Externe Verweise
Dieser Effekt wurde in Anlehnung an das in der DirectX-SDK 2008 (August 2008) beschriebene HDR-Lightning HLSL-Example in GLSL umgesetzt.

Beschreibung
Ist der HDR Effekt aktiv, wird die gesamte Szene in eine Textur gerendert, aus der im nächsten Schritt ein Durchschnitt-Beleuchtungswert ermittelt wird. Dazu wird zuerst eine herunterskalierte Szene-Textur erstellt, welche die Durchschnitte des Logarithmus der Lumince jedes Scenebereiches abbildet. Diese Textur wird iterativ in 4x4-Blöcken herunterskaliert, und in eine finale 1x1-Textur die aktuelle Durchschnittsbeleuchtung der Szene gespeichert. (Von dem schließlichen Durchschnitts-Wert aller Log-Luminances muss dazu noch der Inverse Logarithmus genommen werden).
Ein simulierter adaptiver Iris-Anpassungsmechanismus (`HDRAdaptedLuminance.psh`) nähert dann Frame für Frame in einer Exponentialkurve deinen aktuellen adaptiven Beleuchtungswert an den aktuell berechneten Durchschnittsbeleuchtungswert der Szene an.
Auf die skalierte Version der Szenentextur wird zusätzlich noch ein Bright-Pass-Filter angewendet. Auf die Ergebnistextur wird mittels eines 15x15-Gauss-Kernels ein Bluring angewandt (`HDRGaussBlur15.psh`) welches eine Bloom-Textur erzeugt.
Im letzten Pass (`HDRFinalSceneCompositing.psh`) wird, abhängig von der aktuelen Lichtadaption der Iris, ein Tonemapping auf die HDR-Werte angewandt. Auf die daraus resultierenden LDR-Werte wird per additivem Blending die Bloom-Textur überlagert, was zum finalen Output dieses Effekts führt.
Mittels der Taste F3 lässt sich während des Spiels eine HDR-Effekt-Visualisierungsansicht einschalten:

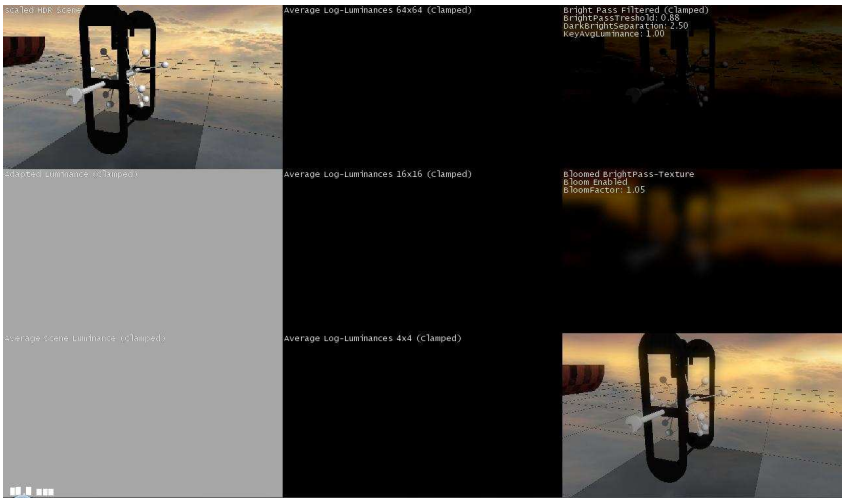


Abb. 1: HDR Effect Chain

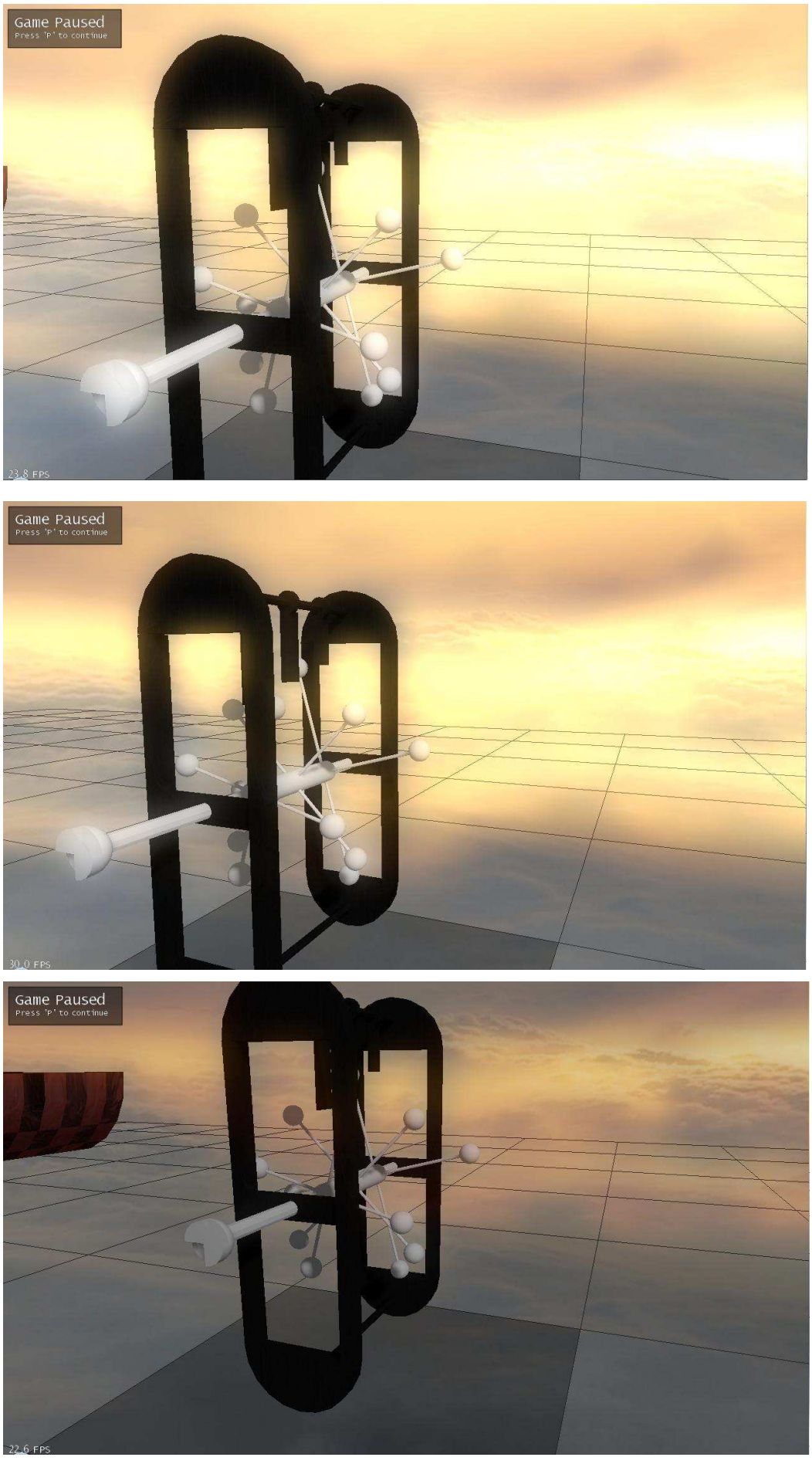


Abb. 2: Unterschiedliche Lichtadaptionen einer HDR-Szene

3. PARTIKELEFFEKTE

Code-Referenzen
Klassen: ParticleSystem
Shader: Particle3DMaterial.vsh, ParticleBillboard.vsh, AlphaTexturedColor.psh

Beschreibung
Partikelsysteme kommen im Spiel an zwei Stellen und in zwei Ausprägungen vor. Werden Blöcke mittels eines Bomb-Specialitems weggesprengt, wird ein einfaches Explosionspartikelsystem (Lineare Translation) aus den Vertexdaten der zu sprengenden Geometrie erzeugt. Die Partikel selbst stellen Billboard-Quads mit einer Alpha-Textur dar.
Etwas gegen die Definition eines Partikels kommen auch 3D-Partikel vor, wobei jedes Partikel eine simple 3D-Geometrie besitzt. Diese sind in der aktuellen Implementierung etwas performancelastiger. Sie werden dann verwendet, wenn die Kugel zum letzten Mal über ein Pfadsegment rollt, was zur Zerstörung des Segments führt. Das Partikelsystem simuliert dann einzelne Brocken bzw. Trümmer, die in den Abgrund rieseln.

Abb. : Billboard-Partikel

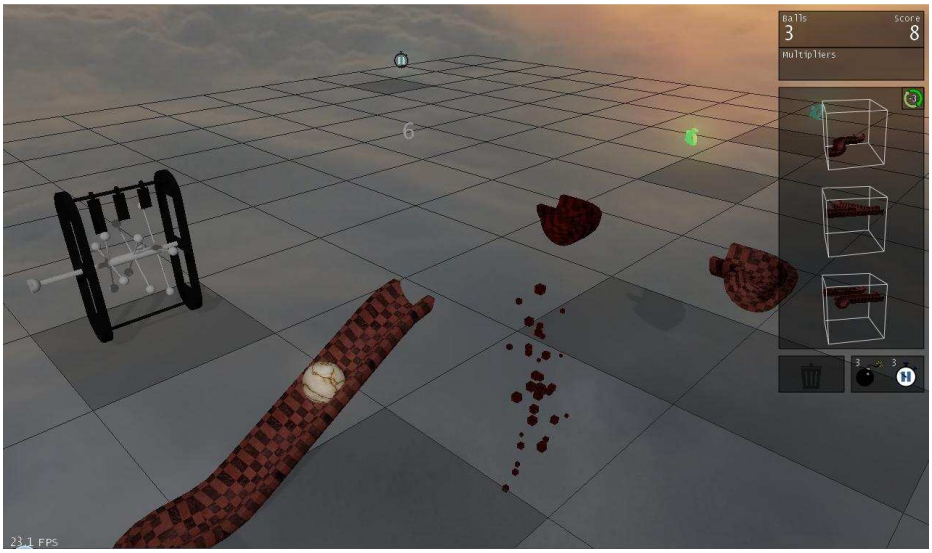
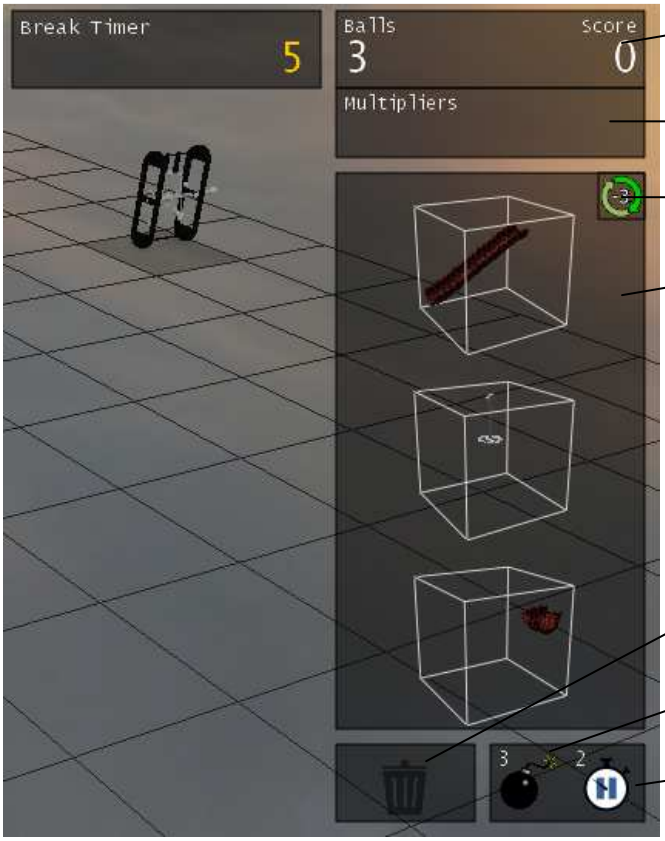


Abb. : 3D-Partikel

Spielanleitung



Anzahl Bälle und Aktueller Score

Aktuelle Multiplier

Refresh NextBlock-List (Taste R)

NextBlock-List

Mülltonne

Bomb-Items

BreakTimer-Items

Hinweis: Ein detailliertes Manual zum Spiel ist sich zeitlich leider nicht zur Gänze ausgegangen und wird nachgereicht. Infos zur Spielsteuerung sowie HDR-Einstellungen finden sich im Menü des Spiels unter dem Menüpunkt Help.