Allgemein

Alle verwendeten shader befinden sich im Shader Ordner

Übung 1 – Prozedurale Geometrie

Am Anfang wird eine 3d Textur erzeugt und Schicht für Schicht mit „Density“-Werten gefüllt. Als Regeln wurde „Shelves“, „Helix“, „Center Hole“ und „Bounds“ verwendet. Die verwendeten Shader sind „texture3d.*“. Dieser kann über die „baseDensity“ uniform beinflusst werden. Diese ist mittels der „+“ bzw. „-“-Taste höher bzw. niedriger gestellt werden.

Danach wird aus der „Density“-Texture eine Geometrie mittels des Marching Cubes Algorithmus erstellt. Die Implementierung orientiert sich an folgender Quelle:
http://paulbourke.net/geometry/polygonise/
Die verwendeten Shader sind „mc.*“.

Die erzeugte Geometrie wird mittles Planar Mapping von der „y“-Richtung aus texturiert. Der verwendeten shader sind „rock.*“

Übung 2 – Parallax mapping

Das Parallax Mapping wurde nur an der generierten Geometrie angewendet und benutzt ebenfalls Planar Mapping von der „y-Richtung“ aus. Die Anzahl der Main Steps und Refinement Steps zum finden des UV-Offsets können mittel der Tasten 9/0 (Main) und 7/8 (Refinement) eingestellt werden. Die Stärke des Effekts ist mittels der Tasten 5 und 6 einstellbar. Die Implementierung befindet sich im shader „rock.frag“.

Übung 3 – Particle system

Ein Particle System kontrolliert alle Particle eines bestimmten Typs. Wenn Particle dazugefügt werden, werden diese in einen Buffer kopiert. Die Daten eines Particles umfassen dessen Position, die Geschwindigkeit und verbleibende Lebenszeit. Der C++ teil des Particles Systems befindet sich in „ParticleSystem.*“

Das Updaten dieser Werte erfolgt mittels eines Vertexshaders, welcher die Positionen ändert und einem Geometry Shader, der Particles dessen Lebenszeit abgelaufen ist löscht. Der Update prozess verwendet ein double buffering system. Mittels Transform Feedback werden die neuen Particles wieder in einen Output Buffer geschrieben, welcher später mit dem Input Buffer getauscht wird. Die Shader zum updaten heißen „particleUpdate#.*“. Die Update Rate kann mittels der Tasten „U“ und „I“ konfiguriert werden.

Gerendert werden die Particles mittels Instanced Drawing. Dieser Shader erhält sowohl Meshdaten als auch Particledaten. Während der Shader für jeden Vertex neue Meshdaten erhält, hat besitzt dieselbe Instances immer diesselben Particledaten. Die wurde mittels „`glVertexAttribDivisor`“ implementiert. Die Shader zum Rendern heißen „particleDraw#.*“.

Der Raycast zum finden der Spawnposition von Particles benutzt einen KdTree welcher auf die erzeugte Geometrie angewendet wurde. Die Spawnpostion befindet sich an der nähesten Stelle unter dem Mauszeiger. Der gespawnte Particletyp wird für jedes spawnen durchrotiert. Der Raycast und Particle Spawn wird mittles der „R“ taste ausgelöst. Die Anzahl der gespawnten Particle kann mittels der Tasten „O“ und „P“ konfiguriert werden.

Übung 4 – Soft shadows

Die Soft Shadows wurden mittels Exponential Shadow Maps (ESM) implementiert. Statt dem Z-Wert wird stattdessen e^(Z * K) in die Shadowmap gespeichert. Der Benutzte Shader ist „esmShadowmap.*“.

Danach wird die Shadowmap vertikal geblurred und in eine Temporäre Textur gespeichert, dann wird diese temporäre Textur horizontal geblurred und die Ursprungsshadowmap mit diesem Ergebniss überschrieben. Der Blurradius ist mittels den Tasten „G“ und „H“ konfigurierbar. Die Sampleanzahl bleibt jedoch unverändert. C++ code ist in „GausFilterer.*“ und Shadercode in „blurShader.*“.

Die Schattenstärke im beim Finalen Shader errechnet sich aus dem Produkt mit dem Wert aus der shadowmap und e^(-k * d), wobei d für den momentanen Abstand zur Lichtquelle steht. Der k wert ist mittels den Tasten X und Y einstellbar. Je höher der K wert desto besser das Ergebnis. Allerdings funktioniert ESM, ab einem gewissen k Wert nicht mehr, da die Wert zu groß werden und außerhalb des durch floats darstellbaren Bereiches liegen.

Die Framerateanzeige kann mittels der Taste „T“ ein und ausgeschaltet werden.

Übung 5 – Tesselation

Es wurde ein durch eine Heightmap erstelltes Terrain implementiert. Das Terrain startet als ein Quad, welches Tesseliert wird. Die Inner und Outer Tesselation hängt von einem Qualitätsmodifier ab, welcher durch die Tasten „N“ und „M“ einstellbar ist. Im Tesselation Evaluation Shader wird die Vertex Position, abhängig von einer Bumpmap entlang der Ebenennormalen verschoben. Die Stärke der Verschiebung kann durch die Tasten „B“ und „V“ beeinflusst werden. Im Geometry Shader werden die Normalen für jedes Dreieck errechnet, welche für die spätere Beleuchtung erforderlich sind. Der Fragmentshader ist ein ganz normaler Phong Shader. Die Shader heißen „terrain.*“.

Mittels der Beistrich Taste kann zwischen dem Normalen Zeichenmodus und dem Wireframemodus hin und her gewechselt werden.