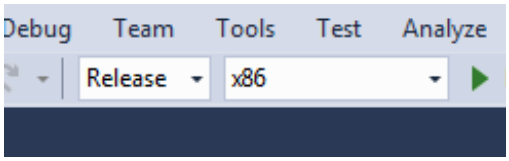
Die Aufgaben dienen der Wiederholung des Inhalts von CG1, und werden in der ersten Einheit behandelt. Es handelt sich nicht um Hausübungen.

Stellen Sie die Visual Studio Build Configuration vorm Erstellen auf jeden Fall auf 32 Bit um, die externen Libraries sind nicht für 64 Bit ausgelegt. Weiters sollten Sie einen Release Build stellen, das ist jedoch nicht zwingend nötig.



Philipp Welsch wird Ihnen im Tutorium mehr zum Umgang mit der Mathematik Library GLM erzählen. Von allen 3D Party Libs ist dies diejenige, die Sie am öftesten brauchen werden. Für diese Übungen brauchen wir die Library auch schon, aber nur in sehr begrenztem Umfang.

Aufgabe 1 - Projekt: ex1_ogre

Wenn Sie das Programm starten, sollten Sie in weißer Farbe auf schwarzen Hintergrund Geometrie sehen. Sie können die Shader übrigens neu laden, ohne das Programm neu starten zu müssen, indem Sie die Space-Taste drücken.

- Im C++ Code sind Matrizen für eine perspektivische Kamera definiert - diese soll auf die Geometrie angewendet werden. Sie müssen dafür sowohl im C++ Code als auch im Vertex Shader Änderungen vornehmen. Sehen Sie sich an, welche Funktionen die Hilfsklasse zum Setzen von Uniforms hat. Wenn Alles klappt, können Sie das Objekt auch mit Hilfe der Maus rotieren.

- Im Moment ist die Geometrie einfärbig. Benutzen Sie zunächst einmal *gl_FragCoord.z*, um das Modell nach Kameradistanz einzufärben: Fragmente, die weiter weg von der Kamera sind, sollen dunkler sein. Sobald Sie das tun, werden Sie feststellen, dass die Geometrie nicht korrekt gerendert wird – das liegt daran, dass ein bestimmter OpenGL State nicht korrekt gesetzt ist. Fixen Sie den entsprechenden State.

- Um Phong Shading zu berechnen, brauchen Sie die Normalvektoren. Das 3D Modell hat Normalvektoren; Diese sind an die *Attribute Location 1* gebunden. (= Default in dieser LVA). Transformieren Sie die Normalvektoren in einen geeigneten Koordinatenraum und geben Sie sie als Farbwert aus.

- Weiters sind im C++ Code Parameter für eine Punktlichtquelle definiert. Übergeben Sie diese an das Glsl Programm, und berechnen Sie Phong Shading. Zur Wiederholung:

  $(k_S * \max(\text{dot}(R, V), 0)^{\text{shininess}} + k_D * \max(\text{dot}(N, L), 0)) * L_{in}$

  Den ambient Anteil lassen wir weg, und die Materialkonstanten k_S / k_D können Sie hardcoden.

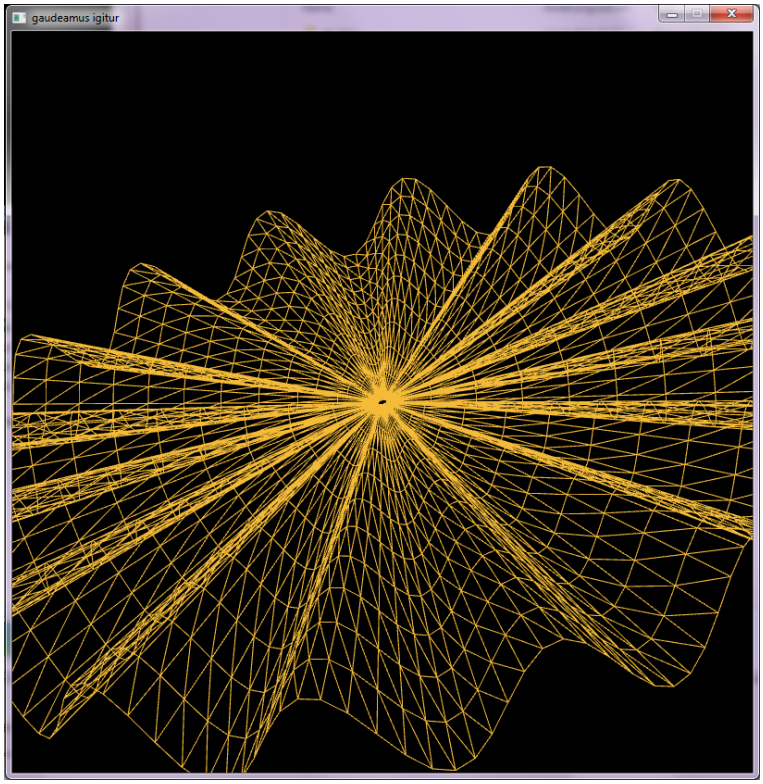
Aufgabe 2 - Projekt: ex2_rings

Im Code läuft ein SFML Timer (sf::Clock) mit, und die Millisekunden seit Programmbeginn werden bereits an den Shader geschickt, dadurch ergibt sich die Farbänderung.

- Ändern Sie den C++ Code erstens wieder so, dass die Kamera korrekt auf die Geometrie angewendet wird.

- Rechnen Sie im Vertex Shader die x/y Koordinaten der Vertices in *Polarkoordinaten* (r, phi) um. Benutzen Sie die GLSL Funktion atan(y, x) um den Winkel phi auszurechnen, und transformieren Sie den Winkel dann ins Interval [0, 1]. Um sicherzustellen, dass Sie ein vernünftiges Ergebnis berechnen, sollten Sie diesen Wert als Farbe ausgeben – Sie sollten einen Kreisförmigen Farbverlauf von Schwarz nach Weiß sehen.

- Benutzen Sie dann diesen Wert in Kombination mit der Zeit als Phasenverschiebung, um die Vertices wellenförmig entlang der Z-Koordinate zu verändern. Benutzen Sie den den Radius r der zuvor berechneten Polarkoordinaten, um das Displacement zum Zentrum hin abzuschwächen. (Siehe Grafik)



- Im C++ Code werden ausserdem zwei 2D Texturen geladen. Binden Sie beide Texturen, und benutzen Sie die UV Koordinaten , welche Ihnen auf der *Attribute Location 4* zur Verfügung stehen, um die Texturwerte im Fragment Shader auszulesen. Setzen Sie den Wert der ersten Textur als Farbwert; Die zweite Textur speichert einen Graustufenwert. Fügen Sie dem C++ Code eine über die Tastatur verstellbare Threshold Variable hinzu, und verwerfen Sie alle Fragmente deren Wert unterhalb des Thresholds liegt.

Aufgabe 3 - Projekt: ex3_quad

In diesem Projekt wird ein einzelnes Quad mit den Eckpunkten (-1, -1, 0) (1, -1, 0) (1, 1, 0) (-1, 1, 0) geladen, sowie eine 512x512Textur.

- Schaffen Sie es, dass das Quad mit genau 512x512Pixeln gerendert wird, und lesen Sie die Textur im Fragment Shader aus.
- Stellen Sie dann das Auslesen der Textur so um, dass der GLSL Befehl texelFetch verwendet wird.
- Implementieren Sie einen Sobel-Filter