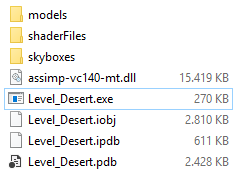
Level\_Desert “Benutzerhandbuch”

Github-Repository:   
 <https://github.com/M1mez/Level_Desert>  
Link auf fertig kompilierte Executables:   
 <https://drive.google.com/open?id=1HNCvg-0us612yfWKfJFn-qSfsWn5aqIr>

Das .exe-file muss im selben Ordner wie die folgenden Dateien/Ordner liegen:

### Steuerung:

Gesteuert werden kann mit den „WASD“- oder den Pfeiltasten.  
Mit „Shift“ kann beschleunigt werden.

### Meshmode:

Durch Gedrückthalten der „Strg“-Taste können die Meshes ohne Textur/Farbfüllung betrachtet werden.

### Levels of Detail:

Die Kakteen haben drei LoDs, welche mithilfe des Meshmodes leicht beobachtet werden kann.

### Glaswände:

Am Rand des Spielfeldes befinden sich transparente Wände, durch die man die Skybox verzerrt wahrnehmen kann. Dadurch, dass man von außen nicht nach innen sehen muss, wurde diese Option nicht implementiert, die Transparenz ist in jedwede Richtung gegeben.

### Scrolling:

Per Drehen des Mausrades kann gezoomt werden.

# Code:

Aufgrund der Absicht, ein vollkommen neues Projekt basierend auf OpenGL 3.3+ zu erstellen, wurde das alte Projekt vom 4. Semester NICHT übernommen und nach einem Tutorial von „[learnopengl.com](https://learnopengl.com/)“ gearbeitet. Die Klassen, die übernommen wurden, wurden studiert und verstanden.  
Aufgrund der verständlichen Struktur wurde diese auch nicht weiter verändert, da sie genau das machen, was vollbracht werden soll.

Modelloading:  
Das Modelloading von komplexen Modellen wird mithilfe der externen Library „[Assimp](http://www.assimp.org/)“ bewerkstelligt, einfache Meshes wie die Ebene, Glaswände und Skybox werden über eigene Methoden erzeugt und gezeichnet.

Die Klassen „Model“ und „Mesh“ wurden mithilfe des Tutorials erstellt, sie enthalten auch nur variablen wie unsigned ints für die Shaderprogram-IDs und Bufferobjects und leicht verständliche Methoden zum Laden der Meshes mittels Assimp und Texturen mittels „[stb\_image](https://github.com/nothings/stb/blob/master/stb_image.h)“.

Die Location der Models wird mithilfe einer Hashmap geladen, deren Keys Teil eines Model-Enums sind und die Values den relativen Pfad zur Executable. Die zuständigen Maps werden zu Beginn des Programms initialisiert.

#### Camera:

Die Camera-Klasse enthält positionsbestimmende Methoden und Variablen und verarbeitet Informationen zum Bewegen und Rotieren des Blickwinkels.

#### Shader:

Die Shader werden mithilfe einer Shaderklasse geladen und die Konstruktoren mittels einer Hashmap, deren Key Shader-Enums sind, aufgerufen.