186.140 Echtzeitgraphik

King's Row

Rebeka Koszticsak, 1325492
J. Sebastian Kirchner, 0926076

Implementierung

Die Basis vom Code ist das CGUE SS15 Projekt von Rebeka Koszticsak und Peter Pollak.

Rendering: Mit dem Rendering zusammenhängende OpenGL Funktionen sind in der Renderer Klasse gesammelt. Bei der Initialisierung der Klasse wird das benutzte OpenGL Profile (3.3) festgelegt. Objekte sind in der MeshNode Klasse gekapselt und enthalten sowohl alle Objektdaten (Vertices, Texturen, …), sowie Buffer Informationen für die Daten auf der Grafikkarte. Durch eine prepare Funktion werden diese Daten der Renderpipeline übergeben und auf die Grafikkarte geladen. Zusätzlich werden diese Objekte in einem Array gesammelt, die in der Rendering Schleife durchiteriert wird. In der Schleife wird die draw Funktion jedes Objektes aufgerufen, sowie die view und projection matrix aktualisiert. Daraufhin übergibt sich das Objekt, zusammen mit Informationen über die Position, selbst, an die draw Funktion der Renderer Klasse, welche den richtigen Shader und Buffer aktiviert, zeichnet, und wieder deaktiviert. Sobald alle Objekte gezeichnet wurden, werden diese durch glfwSwapBuffers auf den Bildschirm geladen.

Beleuchtung: Die Shading ist sehr modular implementiert. Wenn die Objekte geladen werden, wird auch ihr Shader festgelegt. Es ist auch möglich, dass unterschiedliche Objekte, unterschiedlich geshadet werden. In unserem Prototyp wird derzeit für jedes Objekt ein Blinn-Phong Shader verwendet, der Directional-, Spot-, und Pointlights unterstützt. Die MeshNode Klasse speichert das benutzte Shader-Program (Klasse ShaderProgram), welches erzeugt wird, wenn das erste Objekt geladen wird, und für weitere Objekte wiederverwendet wird. Während dem Rendern wird immer der Shader des aktuellen Objektes aktiv gestellt. Der Renderer lässt das Shader Program die benötigte Uniforms befüllen, wozu Daten aus dem Objekt, sowie die aktiven Lichter der Szene benötigt werden.

Texturen: Ähnlich zum Shader, wird die Textur des Objektes beim Object Loading festgelegt und in der MeshNode Klasse mitgespeichert. Zur Initialisierung eines Objektes wird der Pfad der Textur benötigt. Dadurch kann die Klasse Texture initialisiert werden, deren Aufgaben ist die Textur zu laden. Wenn die Textur während dem shading benötigt wird, kann die ShaderProgram Klasse auf diese Informationen durch die MeshNode Klasse zugreifen.

Camera: Die Klasse CameraNode steuert grundsätzliche Kameraeigenschaften wie die View- und Projection-Matrix. Kameras werden in der Main Klasse in einer map gespeichert und können aktiviert werden, in dem sie als die activeCamera Pointervariable gesetzt

werden. Die aktuelle Szene enthält jedoch nur eine Kamera, welche an derselben TransformNode wie die PlayerNode hängt. Die Kamera selbst ist statisch und kann ihre Position nicht ändern, die PlayerNode welche sich die TransformNode mit der Kamera teilt, kann diese jedoch verändern. Vor jedem draw call wird der Input abgefragt und beim updaten der Szene an jede Node des Szenegraphens übergeben, die PlayerNode ändert ihre TransformNode abhängig vom Input, wodurch auch die Kamera Position und Orientation geändert wird.

Szene: Die Objekte werden im Szenengraphen(SceneNode) verwaltet. Jedes Objekt (Mesh, Kamera, Licht) muss im Szenengraph enthalten sein. Jeder Objekttyp hat eine entsprechende Klasse welche beim Laden der Szene bekannt sein muss, von SceneNode erbt, in den Szenengraphen integriert werden kann und typenspezifische Funktionen Implementiert. Zum Beispiel verwaltet die CameraNode die View- und Projection-Matrix, die PlayerNode reagiert auf Input, die MeshNode enthält alle nötigen Informationen zum Rendern wie die Speicherposition auf der Grafikkarte, Shader, Texturen usw. Die Position der Objekte in der Szene werden durch TransformNodes bestimmt an welche die Objekte angehängt werden. Zu der vollständigen Initialisierung müssen alle Knoten verknüpft werden, wobei Ringe im Graphen nicht zulässig sind und der einzige Knoten, keinen Parent hat, die RootNode ist von welcher die komplette Szene ausgeht.

Zusätzliche Libs

- Image Loading: std_image (https://github.com/nothings/stb/blob/master/stb_image.h)
- Object Loading: assimp (http://assimp.sourceforge.net/)
- Glad (http://glad.dav1d.de/)
- GLFW (http://www.glfw.org/)
- GLM (https://glm.g-truc.net/0.9.8/index.html)
- https://www.opengl.org/

Grafikkarte

Der Code ist auf NVIDIA implementiert.

Steuerung

Im Prototyp ist zurzeit klassische WASD Steuerung Implementiert, wobei die Orientierung durch Mausbewegungen gesetzt wird. Um die Applikation zu schließen muss man die Taste Esc drücken.