Lukas Hinterleitner, Daniel Nepp - CGE

Foveated Rendering

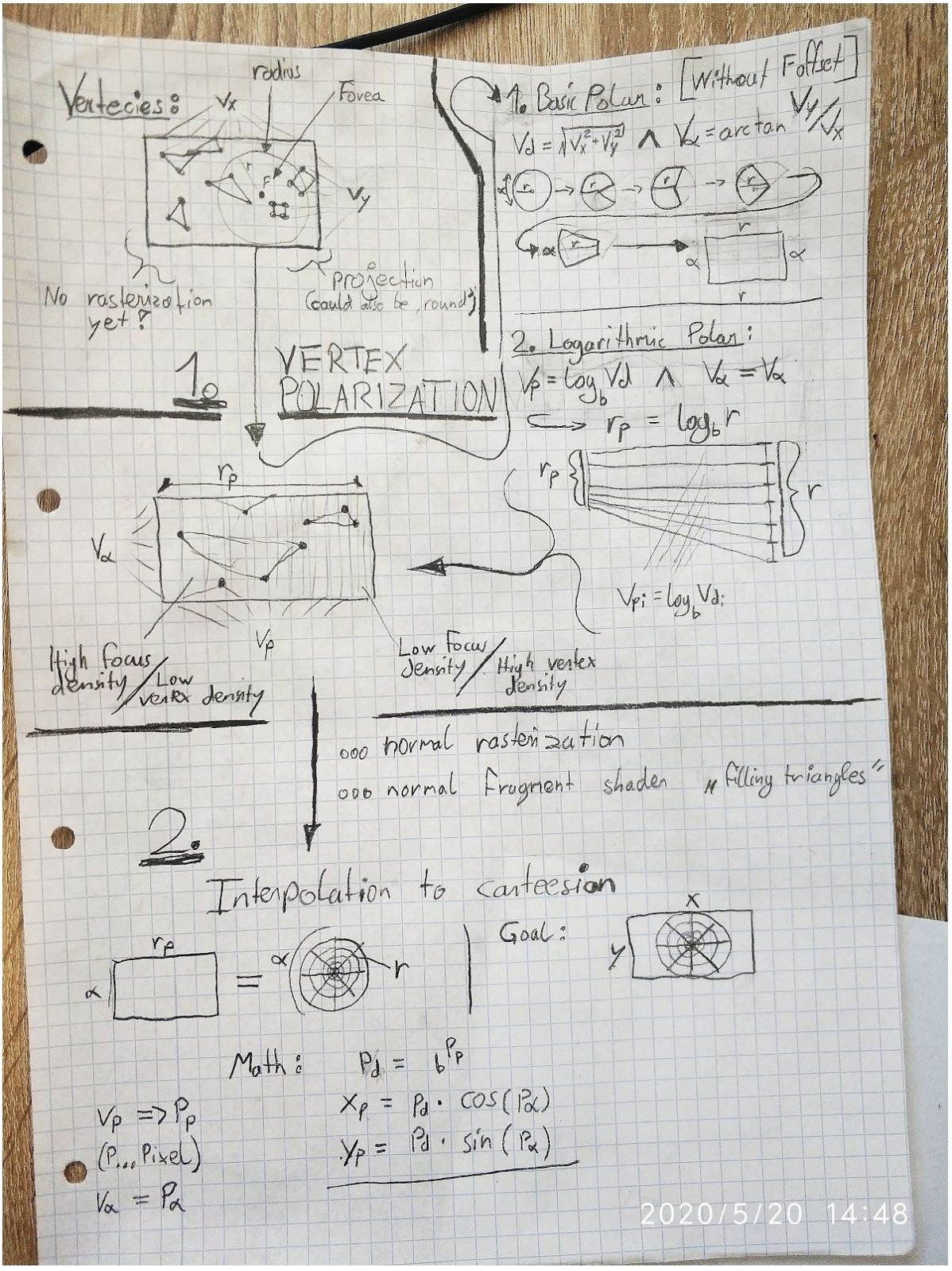
(Fixed Foveated Rendering)

Allgemein

In diesem Projekt haben wir versucht eine einfache Foveated-Rendering Implementierung zu erzielen, indem wir Szenen Vertices in das logarithmisch polare Koordinatensystem transformieren und dann die gerenderten Pixel mittels Fragment Shader wieder in das kartesische Koordinatensystem zurückinterpolieren. Dabei war das Ziel, dass sowohl Fokuspunkt als auch Sichtfeld Radius vom Benutzer angepasst werden können.
Traditionell geht diese Rendering Technik mit dem Eyetracking einher, da wir allerdings sowohl Zeit als auch Ressourcen für eine solche Herangehensweise im Rahmen der Lehrveranstaltung nicht zur Verfügung hatten, haben wir eine Fixed-Foveated-Rendering Implementierung gewählt.
Nicht-Ziel war die Gestaltung komplexer Inhalte wie Modelle und Grafiken. Die Szene sollte dabei nur dazu dienen den Effekt des Foveated-Rendering zu veranschaulichen. Im nachfolgenden Bild ist ein Beispiel, wie so eine vollständig transformierte Grafik aussehen sollte.

Notizen

In der nachfolgenden Darstellung sind ein paar Notizen unsererseits abgebildet, um die Transformation zu visualisieren.



Requirements

Hardware

Grafikkarte: Integrierte oder dedizierte Grafikkarte mit OpenGL 3.3 oder höher

Prozessor: Alle gängigen Prozessoren von Intel oder AMD reichen vollkommen

Speicherplatz: mindestens 20 MB freier Speicherplatz für die Applikation inkl. Build Dateien

Software

Betriebssystem: Windows 10

Libraries: Sämtliche Libraries um OpenGL sind bereits in einem Dependencies Ordner enthalten, daher sind keine weiteren Bibliotheken notwendig

Features

- Einfache Szene als Grundlage umm die Effekte des foveated renderings zu veranschaulichen.

- Simple Maus & Keyboard Steuerung um die Szene perspektivisch zu manipulieren und die Auswirkungen des renderings in den neuen Perspektiven betrachten zu können.

- Der Fokuspunkt (welche standardmäßig in der Bildschirmmitte liegt) kann mithilfe der Pfeiltasten umpositioniert werden um den rendering Effekt an unterschiedlichen stellen zu erreichen. (Simuliert das eye tracking!)

- Der Radius des Sichtfeldes kann mittels der [+] und [-] Tasten vergrößert und verkleinert werden.
  (Radius wird zwar an die Shader übergeben, aber hat im aktuellen Stand des Projektes noch keine Auswirkung)

- Zwei Shader Programmer um einerseits eine log-polar Transformation der Szenen Vertices in einen Texture Buffer zu rendern…
  ...und anschließend die Pixel aus dem Textur Buffer wieder in das kartesische Koordinatensystem zu interpolieren und auf den Bildschirm zu rendern.

Repository

Link zum GitHub Repository: https://github.com/Gleethos/Fovea

Quellen

Der konzeptionelle und mathematische Hintergrund für dieses Projekt stammt aus folgender Veröffentlichung:
https://duruofei.com/papers/Meng_KernelFoveatedRendering_I3D2018.pdf

Implementiert wurde der Algorithmus mithilfe folgender Tutorials:
https://learnopengl.com/Advanced-Lighting/Bloom
https://www.lighthouse3d.com/tutorials/glsl-tutorial/texture-coordinates/