

Administrative	
<b>Group Name / Game Name</b>	Kekara's Express
<b>GitHub Link</b>	<a href="https://github.com/ManuelKeilman/cgue21-KekarasExpress">https://github.com/ManuelKeilman/cgue21-KekarasExpress</a>
<b>Students</b>	Manuel Keilman, 11824531 Ahmed El Agrod, 11811340
<b>Genre</b>	Racing Game
<b>Goal</b>	Deliver your package before you run out of power or time!
Gameplay	
<b>3D-Geometry</b>	Wir importieren so gut wie alles von Blender mittels Assimp. Hierbei haben wir nicht nur einfache Geometrien sondern auch die ganze Strecke, Bäume, Steine, Leitkegel(Hütchen) und Häuser.
<b>Playable/Advanced Gameplay</b>	Ein Auto kann gesteuert werden. Mit dem können Objekte gesammelt werden und das Auto interagiert mit anderen Objekten. Außerdem gibt es ein Ziel mit Bedingungen.
<b>Min. 60 FPS and Frame-Independence</b>	Dies wird erreicht indem wir die FPS mittels glfwSetFrames() auf 60 setzen und wir berechnen uns die deltaTime die dann für die relevanten Berechnungen berücksichtigt werden um Frame-Independancy zu erreichen.
<b>Win/Lose Condition</b>	Ist man nicht im DebugMode so verliert man sobald die Power oder die Zeit negativ ist. Liefert man alle Pakete bevor eine der beiden Fälle auftritt hat man gewonnen.
<b>Intuitive Controls</b>	So wie in vielen Games steuert man das Auto mit WASD und das wird mittels polling implementiert.
<b>Intuitive Camera</b>	Die Kamera kann gesteuert werden um die Umgebung zu untersuchen. Die Kamera wird fixiert sobald man das Auto steuert.
<b>Illumination Model</b>	Wir haben ein directional light als Lichtquelle und jedes Objekt hat normals und ein Material.
<b>Textures</b>	Da wir so gut wie alle Modelle importieren haben, haben alle eine Textur, die mit importiert wird. Verwendet bei allen Objekten außer „FinishLines/DeliveryLines“.
<b>Moving Objects</b>	Wir haben ein Auto das sich bewegt, Cones die sich bei einer Kollision bewegen und Batterien die sich rotieren.
<b>Documentation</b>	Ist vorhanden
<b>Adjustable Parameters</b>	Die angegebenen Aspekte können in dem settings.txt file geändert werden. Somit muss das Programm nicht nochmal kompiliert werden.

## Optional Gameplay

<b>View-Frustum Culling</b>	Unser View-Frustum Culling basiert auf den Clip Space Approach des unten genannten Links [1]. Die Implementation ist in der Klasse ViewFrustum.cpp/hpp zu finden. Weiteres erstellen wir Bounding Spheres für Modelle und Geometrien. Die Implementierungen dafür sind in den jeweiligen Klassen zu finden. Zuerst wird der Mittelpunkt des Objekts berechnet und anschließend die längste Distanz zu dem Mittelpunkt als Radius für die Bounding Sphere benutzt. Notiz: Fahrbahn, Häuser, Bäume und Steine sind ein Objekt.
<b>HUD</b>	Das HUD wird mittels Freetype implementiert. Es zeigt die verbleibende Zeit, die Power, die FPS, die Anzahl der gezeichneten Objekte und die Anzahl der gelieferten Pakete an. Implementierung orientiert sich stark an das learnopengl Tutorial [2]. Das HUD ist ein und ausschaltbar und passt sich für alle Auflösung mit demselben Seitenverhältnis an.
<b>Collision Detection und Advanced Physics</b>	Für die beiden Aspekte verwenden wir die Nvidia PhysX Engine 4.1. Um den Aspekt Advanced Physics zu erfüllen haben wir ein „vehicle“ mittels PhysX erstellt. Dabei haben wir uns stark an die Code Snippets orientiert. Diese sind im „vehicle“ Ordner zu finden.

## Effekte

<b>Shadow Map with PCF</b>	Stark orientiert an das learnopengl Tutorial [3] wobei die Ideen für die Behandlung der Artefakte aus dem Tuwel Kurs sind [4]. Verwendet bei allen Objekten außer dem Auto.
<b>CPU Particle System</b>	Idee für die Klassen Particles.cpp/.hpp aus dem learnopengl Tutorial [5][6]. Jedes Particle hat eine Position, ein Lebenszeit, eine Farbe, eine Skalierung. Es werden nur Particles mit einer Lebenszeit größer 0 gezeichnet und in regelmäßigen Abständen erhalten Particles mit einer negativen Lebenszeit wieder eine positive Lebenszeit. Die genaue Implementierung ist in den vorhin genannten Klassen zu finden. Mit diesem Particle System wird versucht Schnee zu simulieren.
<b>Hierarchical Animation</b>	Reifen werden als eigene Meshes importiert. Diese rotieren und drehen sich so wie das PhysX Objekt.
<b>Environment Map</b>	Es wird eine Cubemap erstellt. Code orientiert sich stark ans learnopengl Tutorial [7] und ist in der Main.cpp zu finden. Verwendet bei „FinishLines/DeliveryLines“, Batterie-Items und Cones.

<b>Cel Shading</b>	Die Farben werden von rgb in hsv umgewandelt, wobei der „brightness“ Wert angepasst wird, so wie es im Tutorial steht [8]. Wird verwendet bei allen Objekten außer Particles.
<b>Contours via Edge Detection</b>	Es wird ein Framebuffer erstellt, der sowohl eine Farbe als auch eine Tiefentextur speichert. Die Tiefentextur wird benutzt um die Kanten zu markieren. Die markierten Kanten werden anschließend in eine neue Textur gespeichert. Diese wird dann mit der Farbtextur des Framebuffer kombiniert damit man Konturen erstellt. Hierbei wird ein Quad gezeichnet welches mit der gemischten Textur gezeichnet wird. Implementierung ist in Main.cpp zu finden. [9][10][11] Wird nicht bei Particles verwendet.

## Inputs

<b>F1</b>	Toggle Wireframe-Mode
<b>F2</b>	Toggle Backface-Culling
<b>F3</b>	Toggle DebugMode
<b>F4</b>	Toggle Win/Lose Condition
<b>F5/F6</b>	Helligkeit erhöhen/verringern
<b>F7</b>	Toggle Particles (Snow)
<b>F8</b>	Toggle View Frustum Culling
<b>F9</b>	Toggle HUD
<b>ESC</b>	Spiel beenden
<b>Mausrad</b>	Zoom
<b>R</b>	Wechselt „Reverse“-Mode, beim Rückwärtsfahren sieht Kamera nach hinten oder nach vorne, je nachdem ob Reverse-Mode an oder aus ist.
<b>Wenn DebugMode an:</b>	
<b>WASD</b>	Kamera Bewegung
<b>Maus Drag and Drop</b>	Rotation der Kamera
<b>Shift/Leertaste</b>	Kamera hinunter/hinauf bewegen
<b>Wenn DebugMode aus:</b>	
<b>WASD</b>	Auto bewegen
<b>B</b>	Handbremse
<b>Leertaste</b>	Auto wird wieder auf Räder gestellt, falls man seitlich oder auf dem „Rücken“ liegt.

## Zusätzliche Informationen

<b>Verwendete Libraries</b>	PhysX, Assimp, FreeType, glew, glfw, glm(EGC library), stb_image
<b>How to get through the game</b>	Sobald die .exe gestartet wird befindet man sich im „DebugModus“, dies bedeutet, dass du dich mit der Kamera frei bewegen kannst. Weiteres ist die „Win/Lose Condition“ ausgeschaltet(Mit F4 einschaltbar). Sobald man den DebugModus mit F3 beendet, wechselt die Ansicht auf das Auto und die Zeit läuft ab. Das Spiel beginnt. Schaltest du die

	<p>Win/Lose Condition nicht ein, kannst du weder gewinnen noch verlieren, du kannst also das Spiel testen und so lange herumfahren wie es dir Spaß macht. Ist die Win/Lose Condition jedoch eingeschaltet, musst du versuchen so schnell wie möglich an allen Häusern vorbei zu fahren um die Pakete abzuliefern bevor die Zeit um ist. Pass jedoch auf. Abkürzungen über die Wiese solltest du nicht nehmen, da dein Auto über die Wiese viel mehr Energie verliert als wenn du dich auf der Straße befindest. Sollte dir die Energie ausgehen hast du ebenso verloren. Versuch also stets Batterie Items einzusammeln um einen Energie-Boost zu erhalten. Viel Spaß!</p>
--	--

## Quellen

- [1][https://cgvr.informatik.uni-bremen.de/teaching/cg\\_literatur/lighthouse3d\\_view\\_frustum\\_culling/index.html](https://cgvr.informatik.uni-bremen.de/teaching/cg_literatur/lighthouse3d_view_frustum_culling/index.html)
- [2]<https://learnopengl.com/In-Practice/Text-Rendering>
- [3]<https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping>
- [4][https://tuwel.tuwien.ac.at/pluginfile.php/2415211/mod\\_page/content/28/Shadow\\_mapping\\_SS19.pdf](https://tuwel.tuwien.ac.at/pluginfile.php/2415211/mod_page/content/28/Shadow_mapping_SS19.pdf)
- [5]<https://learnopengl.com/In-Practice/2D-Game/Particles>
- [6]<http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>
- [7]<https://learnopengl.com/Advanced-OpenGL/Cubemaps>
- [8][https://tuwel.tuwien.ac.at/pluginfile.php/2415211/mod\\_page/content/28/CelShading\\_SS19.pdf](https://tuwel.tuwien.ac.at/pluginfile.php/2415211/mod_page/content/28/CelShading_SS19.pdf)
- [9] <https://gamedev.stackexchange.com/questions/159585/sobel-edge-detection-on-depth-texture>
- [10] [https://learnopengl.com/code\\_viewer\\_gh.php?code=src/4.advanced\\_opengl/5.1.framebuffers/framebuffers.cpp](https://learnopengl.com/code_viewer_gh.php?code=src/4.advanced_opengl/5.1.framebuffers/framebuffers.cpp)
- [11] <https://learnopengl.com/In-Practice/2D-Game/Postprocessing>