

Mama - Dokumentation

3D Geometrie

Mit Hilfe eines Model- und Texturloaders können Objekte auch mit komplizierten Formen geladen werden. Hierfür wird die Assimp-Library verwendet.

Spielbar

Bewegen mit WASD:

Das Charactermodel, sowie die Kamera bewegen sich in die zugewiesenen Richtungen. Dabei werden in der Playerklasse Position, Rotation, Translation und Eine Matrix gespeichert, die mit Hilfe von Events verändert werden können

Events:

Wird die Maus bewegt oder eine Taste gedrückt, so wird ein Event getriggert, das den

Call prozessiert.

Mausbewegung:

Die integrierte perspektivische Kamera bewegt sich mit dem Spielerobjekt. In der Kamera werden Front, Right, Up, sowie Translations/Rotations-Vektoren upgedatet und prozessiert.

Springen:

Mit der **SPACE**-Taste kann der Bär springen.

Window-Events:

F1 wechselt zwischen Fullscreen und Fenstermodus.

F2 startet das Spiel neu, dabei werden alle Gegenstände zurückgesetzt, wie auch die

Spielercharakterposition und das Leben.

ESC beendet das Spiel.

Sammeln:

An einigen Stellen kann der Bär Futter finden. Sammelt er dieses auf, regeneriert sich

seine Lebensanzeige ein bisschen.

Sonstiges:

Durch die **1** kann man Fallen sichtbar machen, denen man ausweichen muss.

Mit der **2** kann man das Spiel pausieren und wieder aufnehmen.

Mit **M** kann man den Ton aus- und einschalten,

+ und - (sowohl normal als auch Num-Pad) ändert die Volume des Sounds.

Debug Funktionen:

F6 (de)aktiviert einen Cube mit Normal Mapping

F7 (de)aktiviert Vertical Synch

F8 (de)aktiviert Frustum Culling

F9 (de)aktiviert Rendern aller Modelle (Fallen, Essen, Endscreens)

F10 (de)aktiviert Wireframe Modus

F11 (de)aktiviert Freie Kamera Modus

Framerate-Unabhängigkeit

Dazu wird eine Deltatime berechnet. Die Anzahl an Sekunden die es für den vorherigen Frame gebraucht hat werden von der aktuellen Zeit in Sekunden abgezogen.

Die Deltatime wird dann z.B. mit der Playergeschwindigkeit multipliziert.

PhysX addiert jeden Frame die Deltatime zusammen und simuliert alle 16,7ms die Szene.

Win/Lose Condition

Beim Drücken der Taste **1** werden am Boden kleine rote sich drehende Würfel sichtbar. In diese darf der Spieler nicht hineinlaufen. tut er es doch, wird die Lebensanzeige kleiner.

Läuft der Spieler in die Fänge des Menschen, so verliert er sogar noch mehr Gesundheit.

Hat der Spieler kein Leben mehr übrig, wird ein Lose-Screen getriggert. Selbiger erscheint auch, sollte man in ein Loch fallen.

Wenn der Charakter das Mamabär Objekt erreicht, hat man gewonnen und ein Win-Screen erscheint.

Intuitive Controls

In der Game-Loop werden alle Events mittels **"glfwPollEvents"** gecheckt. Die Events befinden sich hier in der dazugehörigen Eventklasse, in der verschiedene Key- und Mouse-Callbacks implementiert sind.

Illumination Model

Mehrere fixe Point-Lights erhellen die Höhle und sind als Lampen seitlich an den Wänden dargestellt. Insg. 8 Lichter werden in Renderer.h erstellt und den entsprechenden Shadern als Attribute mitgegeben.

Textures

Die Texturen werden mit Hilfe der STB IMAGE Library für jedes Objekt geladen. Die Texturkoordinaten werden im Mesh gespeichert und beim Rendern der Meshes als Attribute mitgegeben.

Moving Objects.

Bär:

Der Spieler steuert den Bären durch die Höhle.

Fallen:

Wenn die Fallen sichtbar gemacht werden, rotieren diese.

Mensch:

Ein animierter Mensch läuft in einem Stollen am Start hin und her.

Shadow Mapping mit PCF

Von einer einzigen Lichtquelle wird eine Depth-Map generiert. Jedes Pixel wird mit den Depth-Values verglichen, um herauszufinden ob Pixel x ein Schatten ist oder nicht.

Für das Speichern der Umgebungsdaten gibt es eine Cube-Map, deren generierte Daten

vom Fragment Shader bearbeitet werden.
PCF zeigt die Schatten etwas smoother an.

Bloom

In der Szene werden zuerst die hellen Stellen herausgefiltert und in ein neues FBO gespeichert. Mit Hilfe des Gaussian Blurs wird das Bild horizontal und vertikal getrübt und zum Schluss wieder über die ursprüngliche Szene gelegt.

Normal-Mapping

Aus einer Bump-Map/Normal-Map werden die Normalen zum dazugehörigen Objekt gelesen. Im shader werden dann diese fake Normalen für die Berechnung der Lichtinformation verwendet, sodass ein flaches lowpoly Objekt mit komplexen Strukturen versehen werden kann.

Vertex-Skinning

Ein animiertes Model wird mit der Assimp Library in die Szene geladen. Dabei werden Neben den Informationen für unbewegte Modelle zusätzliche Informationen für Bones, Weights, Dauer und Keyframes der Animation geladen.
Die Transformation der Bones im Bezug auf deren Weights wird im Vertex- bzw. Fragmentshader berechnet.

Interface Elemente

Im Spiel wird eine Lebensleiste angezeigt. Dabei handelt es sich um eine grüne Textur die sich mit der Kamera mitbewegt und je nach Spielerleben die z Achse neu skaliert.

Ist eine Win- oder Lose-Condition erfüllt wird am Bild ein Endscreen angezeigt. Dabei handelt es sich um spezielle Texturen die in diesem Fall geladen werden. Die Kamera positioniert sich dann vor diesen und wird gesperrt, das Spiel pausiert. In diesem Modus kann man entweder mit Enter das Spiel neustarten oder mit X das Spiel beenden.

PhysX Collision Detection

Alle Kollisionen werden von PhysX berechnet. Die gesamte Szene wurde als Triangle Meshes cooked. An gewissen Stellen wurden eigens ungerenderte Collision Wall Objekte benutzt um die Kollisionen besser zu unterstützen. Der Rest wurde aus den gerenderten Objekten generiert.

Physx prüft mit Raycasts entlang der Kamera Sichtlinie ob sich ein Objekt zwischen dem Spielercharakter und der Kamera befindet und die Kamera wird dann je nach gefundener Kollision angepasst. Trigger und der Spielercharakter selbst werden dabei ignoriert.

PhysX Character Controller

Der Spielercharacter wird in PhysX durch einen **PxController** repräsentiert. Der Controller ist das was eigentlich durch den Spieler bewegt wird, während das Charaktermodell sich nur an die Position des Controllers anpasst.

Alle Interaktionen mit Umgebungsobjekten werden durch den Controller ausgelöst.

PhysX Trigger

Alle Fallen, Nahrungsgegenstände, wie auch der Mutterbär und der Cowboy werden in PhysX durch Trigger repräsentiert. Wenn der Controller mit solch einer Triggerzone überlappt wird von PhysX die **onTrigger** Funktion aufgerufen. Das funktioniert indem die von uns erstellte PhysX Klasse **PxSimulationEventCallback** implementiert, wodurch die PhysX Scene dann an sie gewisse Events weiterleitet. Darunter auch wenn ein PhysX Objekt mit einem Trigger überlappt.

PhysX Dynamic Objects

Es gibt in der Spielwelt Kisten und Steine die vom Spieler verschoben werden können. Diese sind in PhysX durch einen **PxRigidDynamic** Actor realisiert. Die Kollisionen werden mit Hilfe der **onCommand** Funktion geprüft und die Spielergeschwindigkeit je nach Masse angepasst.

Frustum-Culling

Hier wird um jedes Model eine Boundary-Box erstellt, die berechnet, ob sich das Objekt innerhalb, oder außerhalb der Camera-LookAt befindet. Danach werden die Eckpunkte der Box mit den 6 Frustum Flächen verglichen. Gibt es eine Fläche von der aus alle Punkte außerhalb liegen wird das Objekt ausgeblendet.