Super Marble Race

Steuerung

Allgemein:

Taste	Effekt
ESC	Spiel schließen
E	Toggle Edit-Mode
F2	Frame Time on/off
F3	Wire Frame on/off
F4	Texture-Smapling-Quality: Nearest/Bilinear
F5	Mip Mapping-Quality: Nearest/Linear
F8	Viewfrustum-Culling on/off (Anzahl der gecullten Tracks warden bei F2 unterhalb angezeigt
F9	Transparency on/off

Edit-Mode:

Taste	Effekt
W,A,S,D	Steuerung der Kamera in x,z-Richtung
Mouse	Steuerung der Richtung der Kamera
SPACE	Kamera nach oben fliegen
SHIFT_LEFT	Kamera nach unten fliegen
CTRL + S	Speichern des aktuellen Tracks
0	Neuen Ring einfügen
1	Neue Rampe einfügen
2	Neue Gerade einfügen
3	Neue Rechtskurve einfügen
4	Neue Linkskurve einfügen
U	Nächstes Streckenteil auswählen (Vibration signalisiert aktives Teil)
I	Nächsten Ring auswählen (Vibration signalisiert aktiven Ring)
T + Numpad(9,8,7,6,5,4)	Translation des ausgewählten Elements: 7 = runter, 9 = hoch, 8,4,5,6 = vor, links, zurück, rechts
R + Numpad(9,8,7,6,5,4)	Rotation des ausgewählten Elements
ENTER	Neustart der Murmel

Race-Mode:

Taste	Effekt
W,A,S,D	Steuerung der Murmel
Mouse	Bewegen der Kamera rund um die Murmel
ENTER	Neustart der Murmel
C	Reset der Bestzeit
B	Instant-Win

Features

- Frei bewegende Kamera: das freie Bewegen der Kamera ist im Edit-Mode möglich
- Bewegende Objekte: die Murmel lässt sich steuern, alle Streckenteile lassen sich verschieben und rotieren
- Texture Mapping: alle Elemente im Spiel haben UV-Koordinaten, die auf eine Textur gemappt angezeigt werden

- Lighting:
    - Es gibt im Spiel ein globales Licht, welches mit Shadow-Mapping Schatten wirft. Dieses Licht befindet sich auf 0,200,0
    - Jedes Streckenelement hat eine Point-Light Lichtquelle, die sich in der y-Achse um +3 verschoben um den Ursprung im Modelspace befindet und dabei rotiert (weiße Kugeln im Spiel)
- Steuerung: frei bewegen im Edit-Mode und Steuerung der Murmel im Race-Mode

Gameplay

Der Spieler muss die Murmel so steuern, sodass er die Ringe einsammelt. Schafft der Spieler das in einer kürzeren Zeit, als die Bestzeit beträgt, so hat er gewonnen, ansonsten hat der Spieler verloren. Mit ENTER kann das Spiel neu gestartet werden.

Komplexe Objekte

Das Spiel besteht aus Streckenteilen. Diese Teile sind beispielsweise Kurven, Geraden und Rampen. Diese Objekte können im Edit-Mode verschoben werden. Weiters gibt es auf der Strecke Ringe platziert. Diese Ringe müssen eingesammelt werden. Als Hintergrund wurde eine Skybox verwendet.

Animierte Objekte

Die Murmel bewegt sich auf der Strecke. Ein Kopf bewegt sich immer zur Bewegungsrichtung der Murmel mit. Damit ist es für den Spieler leichter die Murmel zu steuern.

View-Frustum-Culling

Es wurde ein Frustum Culling mit einer Shere implementiert. (Radius 10 vom Ursprung der Tracks)

Man kann die Anzahl der gecullten Tracks anzeigen indem man F2 drückt (F8 zum aktivieren).

Transparenz

Es wurde eine Transparenz bei dem CPU-basierten Particle-Effekt verwendet. Diese Partikel kann man bei der bewegenden Murmel sehen (es wird Spur gezogen).

Experimentieren mit OpenGl

Es wurden VBOs, VAOs und FBOs verwendet. Die FBOs wurden bei dem Bloom-Effekt verwendet, um den Gaußfilter anwenden zu können.

Effekte

- Normal-Mapping: Für die Streckenteile wurde Normal-Mapping angewendet. Der Algorithmus wurde nach der Theorie von http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/ implementiert.
- Shadow-Mapping: Die Streckenteile werfen einen Schatten. Die Theorie wurde von dem Repitutorium umgesetzt und Verbesserungen wurden nach der Theorie von http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/ eingefügt.
- Bloom: Die Ringe werden mit einem Gaußfilter bearbeitet. Der theoretische Hintergrund wurde von dieser Seite genommen: http://learnopengl.com/#!Advanced-Lighting/Bloom

- CPU-basierte-Particle (Instancing): Bei dem Win-Screen (Taste-B für instant Win) werden Particle mit Instancing generiert. (bei dem ersten Gewinn 5 Sekunden Wartezeit)
  Quelle: http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/

Tools

Für die Erstellung der Objekte wurde Blender verwendet.

Die Normalmapps wurden mit einem Nvidia Plugin für Photoshop generiert (Quelle: https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop)

Verwendete Libraries

- Glew
- Opengl
- Glfw: Für das Öffnen des Windows-Fensters
- FreeImage: Laden der Texturen
- Bullet: Physics-Engine für die Kollision der Murmel mit den Streckenteilen