GLOW

1. Free Movable Camera

Unsere Kamera wird mit der Maus gesteuert. Wohin die Maus bewegt wird, dorthin schaut die Kamera. Die View-Direction wird mittels glm::lookAt(position, direction, up) berechnet. Zusätzlich wird durch die Blickrichtung der Kamera die Richtung der Fortbewegung festgelegt. Gesteuert wird jedoch die Spielfigur, während sich die Kamera bei Fortbewegung der Spielfigur nach dieser richtet.

2. Moving Objects

An die Spielfigur hierarchisch angeschlossen ist die Kamera. (wie eingangs erwähnt)

Einfache bewegte Objekte sind beispielsweise Münzen. Sie sind stationär und rotieren sich ständig. Die Anordnung der Münzen im Korridor wird random generiert, wodurch das Spielsetting bei jedem neuen Spiel ein neues ist.

Hierarchische Strukturen finden sich bei der Spielfigur. Um diese rotieren zwei kleinere Kugeln in konzentrischen Bahnen. Hier wurde die Spielfigur als Parent und die beiden kleinen Kugeln als Child gesehen. Bewegt sich der Parent, so folgen die Child Elemente. Zusätzlich zur Translation und Rotation der Spielfigur, die ihre Matrix an die Child Elemente weitergibt und somit an sich bindet, haben die Child Elemente auch eine eigene Rotationsmatrix. Dadurch rotieren sie eigenständig um die Spielfigur.

3. Texture Mapping

Texture wird mittels FreeImage geladen. Quelle:
http://www.mbsoftworks.sk/index.php?page=tutorials&series=1&tutorial=9

Bei der Verwendung wird die Textur an den Shader übergeben. Jedes Objekt hat eine Textur.

4. Simple lighting and Materials

Wir haben eine Glühbirne implementiert, welche sich oberhalb der Kugel befindet. Sie fliegt mit der Kugel mit. Somit ist es immer nur in der Umgebung der Kugel hell.

Alle Objekte werden beleuchtet.
Die Skybox ist unabhängig von der Lichtquelle und ist daher immer sichtbar.

5. Controls

W = beschleunigen.
Mauskursor = Richtungsänderungen
Leertaste - Boostspeed

6. Gameplay

Ziel des Spieles ist mit der Spielkugel so schnell wie möglich ins Ziel zu kommen. Dabei kann man Punkte in Form von Zeitboni sammeln, indem man Münzen einsammelt. Bei Kollision mit einem Hindernis wird die Zeit auf 0 gesetzt, die Spielfigur an den Start zurück gesetzt und das Spiel beginnt von neuem.

Am Spielende wird die gespielte Zeit ausgegeben. Dabei wird abgeprüft, ob man schneller war als bei einem vorhergehenden Versuch. Ist das der Fall, wird die neue Bestzeit als Referenz für die weiteren Spielversuche gespeichert und herangezogen.

7. Features of the game

Speedboost:
Beim gleichzeitigen Drücken von W und Leertaste.
Außerdem wird hier der Motion-Blur Effekt getriggert. (Siehe Punkt 9 Effekte)

8. Collision Detection

Wir haben eine simple collision detection implementiert. Fliegt die Kugel außerhalb des Korridors oder gegen ein Hindernis, wird sie auf den Startpunkt zurück gesetzt.

Beim Korridor kontrollieren wir, ob die Position des Spielerseinen bestimmten x,y,z Wert überschreitet. Wenn dies der Fall ist, wird der Spieler auf die Anfangsposition gesetzt und das Spiel beginnt von vorne.

Bei den Münzen schauen wir, ob sich die Spielerposition im Würfel der Größe 1x1x1 um die Münzposition befindet. Wenn dies auch der Fall ist, wird die Münze gelöscht und die Anzahl der gesammelten Münzen erhöht sowie ein Zeitbonus gutgeschrieben.

9. Frustum Culling

Ist implementiert. Die Ausgabe der gezeichneten Objekte wird beim Spiel im linken unteren Bildschirmeck angezeigt.

10. Effekte

- Motion Blurr (1.5 Punkte)
  (http://http.developer.nvidia.com/GPUGems3/gpugems3_ch27.html)

- GPU Partikelsystem mit transform feedback (1.0 Punkte)
  http://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/

- Shadow Maps (1.5 Punkte)
  (http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/)

11. Model Creation

Modelle wurden rein über Blender modelliert und mit dem Assimp-Modelloader in die Szene geladen. Die Modelle der Münzen wurde von google-warehouse-3d bezogen und in das benötigte Format gebracht.

12. Externe Libraries

glew32.lib

opengl32.lib

glfw3.lib

FreeImage.lib

assimp.lib

13. Zusätzliche Information:

Die F1 - F9 Tasten sind belegt wie laut Spezifikation gefordert.
Die Tasten 1,2 bzw. 3,4 können zur Steuerung der Helligkeit und der Transparenz verwendet werden.

Alle diese Tasten sind mittels Callback Funktionen implementiert.