# LEGO Dark Age of Legorian

## Grundlegendes Gameplay

Das Ziel unseres Spiels ist es sich mit unserem legendären Helden bis zum Ausgang des Labyrinths durch zu kämpfen. Auf dem Weg dorthin erwarten ihn die dunklen Mächte von Legorian. So muss er sich blutige Kämpfe mit feindlichen Rittern, Trollen, Orks und mit Feuerbällen schießenden Magiern liefern um nicht zu sterben. Da unser Held noch nicht sehr erfahren im Umgang mit dem Schwert ist kann er gegnerische Attacken nur zufällig blocken oder ihnen ausweichen. Vereinzelt finden sich in den Labyrinthen Brunnen welche alle Wunden unseres Helden zu heilen vermögen.

## Effekte

Wir haben 3 Effekte implementiert.

- Partikelsystem (*Particlesystem*): Das Partikelsystem ist bei jedem bösen Zauberer zu finden. Dieser zaubert aus der Ferne Feuerbälle die er unserem Helden entgegen schießt. Die Bälle werden als Partikelsystem realisiert.
  Bei der Initialisierung des Effektes bekommen alle Partikel neue Zufallswerte (Lebensdauer, Richtung). Hat ein Partikel seine Lebensdauer überschritten wird es neu Initialisiert.
  Der Feuerball hat verschiedene Modi. Zu allererst gibt es eine kleine Explosion und der Ball fliegt los. Sollte der Ball eine Wand oder unseren Helden treffen endet er wieder mit einer Explosion. Sollte er im Nichts enden so werden die Partikel einfach nicht mehr erneuert und so verpufft der Feuerball einfach. (Quelle: NeHe Tutorials)
- Wasser(*Waterfield & WaterManager*): Unser Wasser ist in jedem Brunnen im Level zu finden. Alle paar Sekunden fällt ein Tropfen hinein und es entstehen an dieser Position Wellen. Die Position der Tropfen ändern sich jedes Mal. Der Effekt wird durch eine eigens erstellte Height-Map erzielt. Dabei wird in einem bestimmten Toleranzbereich jeweils für X und Z eine Zufallszahl ermittelt, die dann die Position des Wassertropfens im Raum ergibt. Wenn der Tropfen die Wasseroberfläche erreicht hat, wird mittels einer Distanz-Abfrage getestet, welche Punkte im Height-Map-Raster sich im Einzugsbereich (InnerCircle<=Point<=OuterCircle) befinden. Dabei wird auf die sqrt Funktion verzichtet, in dem die Werte quadriert werden. Diese Distanz wird kontinuierlich vergrößert bis die Welle den Rand des Brunnens erreicht und schließlich verschwindet. Kurze Zeit später Beginnt der Vorgang von Vorne. Die Texturanimation wird mittels Spiegelung realisiert.
- Spiegelung (*Waterfield & WaterManager*): Die Spiegelung findet sich direkt beim Wasser. Unser klares reines Wasser Spiegelt seine Umgebung mittels Render to Texture und heilt gleichzeitig unseren Helden. Die Textur wird dabei auf die Vertices der Wasser-Height-Map aufgetragen und ergibt somit das Bild des spiegelnden, sich dynamisch bewegenden Wassers.

## Transparenz

Tranzparenz kommt bei unserem Partikelsystem vor. Die Partikel werden aufsummiert und dadurch entsteht der Feuereffekt.

Kohl Stephanie 0626088 stephanie.kohl@gmx.at 532
Ulrike Zauner 0625479 0625479@gmail.com 532

## View Frustum Culling mit Bounding Boxes

Wir haben das Frustum Culling mittels Bounding Boxes realisiert. Es ist zu finden in *FrustumG*

## Was tun die F-Tasten?

Wir haben die F Tasten wie folgt definiert:

- F1: Frustum Culling ein/aus
- F2: Render Mode ändern
  - Immediate Mode
  - DrawArrays
  - DrawArraysAndVBO
  - DisplayLists
  - GLCommands
- F3 Wire Frame modus ein/aus
- F4 Framerate ausgeben ein/aus

Die aktuellen Einstellungen kann man in der Titelleiste des Spiels sehen. Es gibt auch eine Änderungsausgabe im Debugfenster.

## Animierte Objekte

Unser Held und alle seine Gegner werden als animierte Objekte in unserem Spiel realisiert. Wobei jedes Model mehrere Animationen bietet.

## Steuerung

Die Steuerung unseres Spiels erfolgt wie üblicherweise mit den Tasten W,A,S,D. Mit Space schlägt man zu. Mittels der Pfeiltasten kann die Kamera gedreht werden und mit X und Y in der Höhe verstellt werden.

## Modellierung

Unser Models wurden mit Maya erstellt und mittels Milkshape in das MD2 Format exportiert.

## Features

### Manager

Für alle Objekte die mehrmals ins Spiel geladen werden müssen wie zum Beispiel Monster und Wände haben wir jeweils einen Manager erstellt. Der ModelManager verwaltet z.B. alle benötigten Models (laden, zeichnen), der MonsterManager verwaltet alle unsere Monster und der TextureManager kümmert sich um unsere Texturen.

Kohl Stephanie 0626088 stephanie.kohl@gmx.at 532
Ulrike Zauner 0625479 0625479@gmail.com 532

## ModelLoader

Wir haben einen simplen MD2Modelloader implementiert.

## LevelLoader

Derzeit gibt es 2 Level durch die sich unser Helden kämpfen muss. Weiters haben wir unsere Leveldatei erweitert.
Unser Level wird durch die Klasse *RPGLevel* aus einer Datei geladen und angezeigt. Zum Beispiel sieht ein Level so aus:

```
21 20

10
H L DungeonFloor02
U L mauer2
Z L mauer3
M L mauer1
E L DungeonFloor03
S L DungeonFloor03
W L brunnen
T L torbogen
Q E zauberer
P E ritter
MUZMUZMZUMUZUMUZMUZUM
ZHHHHHHHHHHHHHZHHHHHM
UHHHPHHWHHHPHHTHHHHHU
ZHHHHHHHHHHHHHMHHHHHZ
UZUTUZMZMUZZUMUMZTUMU
MHHHHHHMHHHHHHZHHHHHZ
MHHHHHHTHHHHHHUHHHHHM
ZHHHHHHZHHHHHHMHHHHHU
UMUZMZUMZUMUTMUMZTMUZ
ZHHHHHZZHHHHHHZHHHHHM
UHHHHHTHHHHHHHUHHEHHU
MHHHHHMHHHHHHHMHHHQHM
ZUTMUUMTUMZUMUZUMZMZU
MHHPZHHHHHUHHHZHHHHHM
UHHHUHHHHHTHHHTHHWPHZ
ZHHHMHHHHHMHHHUHHHHHM
MMZTZTZTMTZTMTMTMUZUZ
ZSHHHHHHHHHHHHHHHHHHM
MHHPHHHHHHHHHHHHHHHHU
MUMZMUZMUZUMZZUMUMUZZ
```

Am Anfang wird die Größe unseres Levels angegeben, hier 21 x 20. Danach wieviele verschiedene Objekte geladen werden sollen. Also 10 in unserem Beispiel. Jetzt werden die verschiedenen Objekte welche dann geladen werden angegeben. Der erste Buchstabe definiert den Code des Objektes im Level der Nächste gibt an ob es sich um ein Levelobjekt oder einen Gegner handelt. DungeonFloor02 bezeichnet sowohl den Namen des md2 Models sowie den Namen der zu ladenden Textur. S und E müssen unbedingt angegeben werden. Kommen sie öfter als einmal vor wird eine Fehlermeldung ausgegeben.
Dann wird in Form einer Matrix angegeben wo welches Objekt geladen wird.

## Verschiedene Monster

Wir haben verschiedenste Monster in unserem Spiel. Als Nahkämpfer finden sich Ritter, Goblins und ein Troll. Unser böser Magier ist der einzige Fernkämpfer im Spiel und macht uns das Leben mittels Feuerbällen (siehe Partikelsystem) zur Hölle.

Kohl Stephanie 0626088 stephanie.kohl@gmx.at 532
Ulrike Zauner 0625479 0625479@gmail.com 532

Die Monster in unserem Spiel suchen sich einen Pfad durch das Labyrinth (mit Hilfe des **A*** Algorithmus zu finden in *Pathfinder*) zu der letzten Position des Helden. Haben sie diese erreicht suchen sie sich einen Weg zur momentanen Position des Helden. Unsere Monster können uns auch angreifen und tun dies auch, sobald wir in Ihre Reichweite kommen (auch wenn es seinen Weg noch nicht beendet hat). Sollten wir versuchen wegzulaufen beginnt es uns nach dem bekannten Schema zu verfolgen.

Kohl Stephanie 0626088 stephanie.kohl@gmx.at 532
Ulrike Zauner 0625479 0625479@gmail.com 532