**LANGSTEINER Irene, ERHART Jürgen**

# CAVE-Maze

CAVE-Maze ist ein dreidimensionales Labyrinthspiel für den CAVE oder CurvedScreen. Der Benutzer navigiert durch ein Labyrinth und sammelt, neben der Suche nach dem Ausgang, Gegenstände auf.
Der Benutzer kann mit der Wand durch das Labyrinth navigieren und dabei Gegenstände aufsammeln. Mit dem Joystick kann er nach vorne und hinten gehen und nach rechts und links rotieren. Die mittlere Taste beendet das Programm.
Erreicht der Benutzer den Ausgang (markiert durch eine Scheibe am Boden) und hat alle Gegenstände aufgesammelt, beendet sich die Applikation.

Die Applikation befindet sich im Verzeichnis `/home/vrpk/vrpk23/project/OpenSG` und wird mit `make run-cave-maze` gestartet.

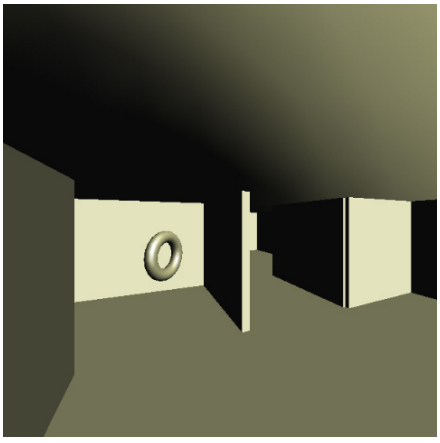
## Dokumentation zur Programmierung

Es wurde mit OpenSG programmiert.
Das Labyrinth besteht aus den Wänden, einer Bodenfläche und einer Höhle, in der sich das ganze befindet. In den Gängen sind Gegenstände, die der Benutzer aufsammeln muss.

`createScenegraph`:
Hier wird das Labyrinth aufgebaut. Wände und Gegenstände werden aus Arrays eingelesen. Mit den Methoden `translate` und `texturize` werden einzelne Teile positioniert und texturiert. Diese Programmierung ermöglicht eine leichte Erweiterung des Labyrinths um neue Wände, Gegenstände, oder ganze Level. Zusätzlich ist es möglich, mittels zweier `define` Anweisungen im Sourcecode einzustellen, ob man das Labyrinth mit Texturen, mit gefärbten Wänden oder völlig ohne Texturierung/Färbung benutzen möchte.

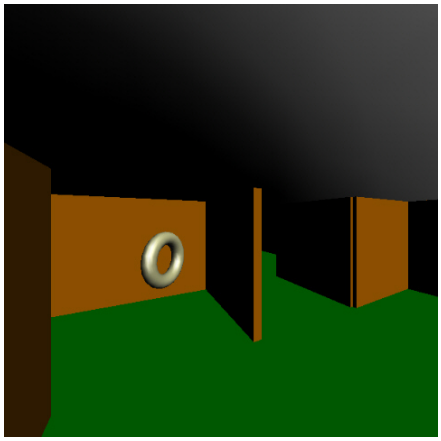
Einfaches Labyrinth:

```
//#define TEX
#define PLAIN
```



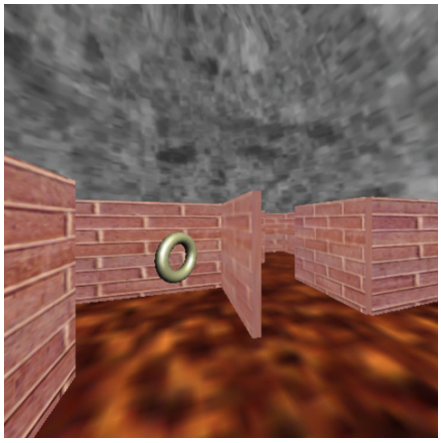
Farbiges Labyrinth:

```
//#define TEX
//#define PLAIN
```



Texturiertes Labyrinth:

```
#define TEX
//#define PLAIN
```



`checkCollectible:`
Diese Methode berechnet für die jeweilige Position und Orientierung des Benutzers, ob er einen Gegenstand einsammeln kann.
Wenn das der Fall ist, wird bei der nächsten Schleife dieser Gegenstand nicht mehr gezeichnet.

`checkExit:`
Hier wird berechnet, ob der Benutzer an das Ende des Labyrinths gekommen ist, und ob er alle Gegenstände aufgesammelt hat.

`stepForward, stepBackward, rotateLeft, rotateRight:`
Das sind die Funktionen, die das Bewegen ermöglichen. Die ersten beiden Funktionen verfügen über eine Collision-Detection, das bedeutet, dass vor jeder Bewegung berechnet wird, ob eine Kollision mit einer Wand bevor steht, wenn in diese Richtung bewegt wird. Dazu wird ein Strahl von der aktuellen Kamerposition in die Szene geschickt und berechnet,

ob er dabei auf eine Wand trifft. Ist das der Fall, und die Entfernung zu dieser getroffenen Wand kleiner, als die Strecke die man sich mit einem Schritt bewegt, bleibt die Position des Spielers gleich, wenn keine Kollision auftritt, werden neue Koordinaten berechnet.

Zu Beginn und zu Testzwecken haben wir das Labyrinth für den Desktop programmiert mit der Steuerung über die Tastatur.

- `w`, `s`, `a`, `d` – Schritt vor, zurück, nach links, nach rechts
- `q`, `e` – Drehung um 90° nach links bzw. rechts

Danach haben wir die Steuerung an die Wand des CAVEs angepasst, sowie den SimpleSceneManager durch den CAVESceneManager ersetzt. Mit dem Joystick kann man sich nach vorne und hinten bewegen und nach rechts und links rotieren. Mit der mittleren Taste kann das Programm beendet werden. Im CAVE rotiert man im Gegensatz zur Desktop Variante nicht um 90°, sondern um 3°. Dies ist nötig um eine sanftere Drehung zu ermöglichen und die Immersion zu erhöhen.

In der Desktop Variante kann man zusätzlich mit der Taste `m` eine Karte des Labyrinths einblenden. Da der CAVESceneManager jedoch die Viewports selbst verwaltet, ist diese Funktion in der CAVE Variante nicht mehr verfügbar.

Dateien:

- cave-maze.cpp – Sourcecodes für die CAVE Variante
- maze.cpp – Sourcecode für die Desktop Variante
- floor.jpg, bricks.jpg, ceiling.jpg – Texturen
- end.jpg – Für die Desktop Variante, erscheint wenn alle Ringe eingesammelt, und der Ausgang gefunden wurde.