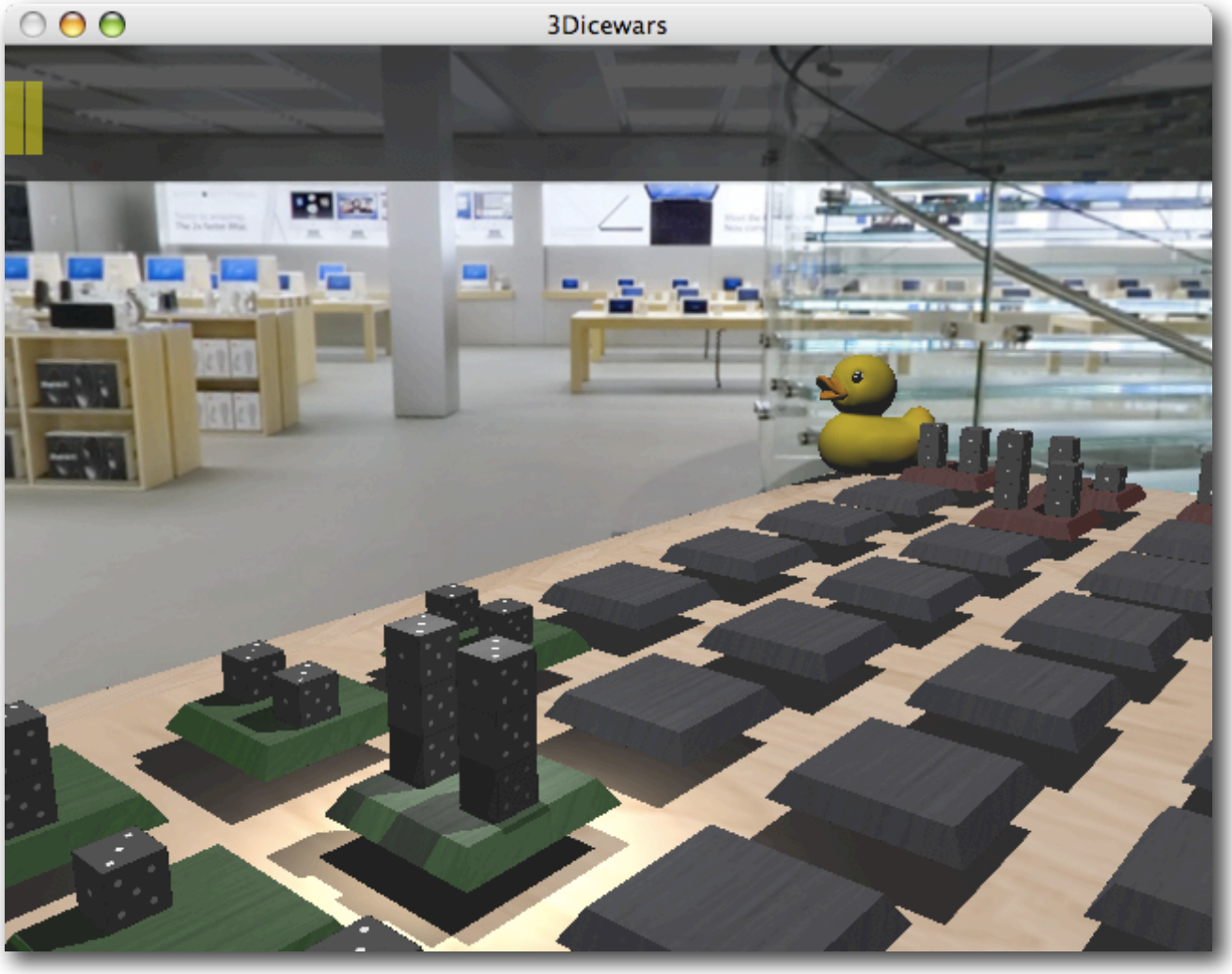
# CG2LU - SS2007

3. Abgabe - Spiel: 3Dicewars

Thomas Zöchling 0026540 935 thomas.zoechling@gmx.at

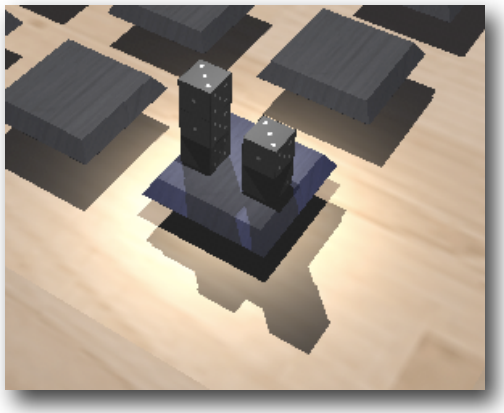


**Gameplay**
Das Spiel orientiert sich an Risiko. Im Gegensatz zu Ländern, gilt es in 3Dicewars allerdings abstrakte Kacheln zu besetzen. Kämpfe werden durch Würfelwahrscheinlichkeiten entschieden. Jeder der 4 Spieler startet mit 5 Würfeln in einer der Ecken des quadratischen Spielfeldes. Der blaue Spieler wird von einem Menschen gesteuert wohingegen die 3 anderen automatisch ziehen. Man kann sich in Einzelfeld Schritten bewegen. Diagonale Züge sind nicht erlaubt. Leere Felder können durch einfaches Darüberfahren eingenommen werden. Der Selektionscursor wird durch ein Spotlight repräsentiert um die Auswahl des gewünschten Würfelstapels zu erleichtern. Besetzte Nachbarfelder können nur durch Kampf eingenommen werden. Dazu zieht man einen eigenen Würfelstoß über den eines Gegners. Das Spiel endet, sobald das gesamte Spielfeld von einem Spieler eingenommen wurde. Nach jeder Runde erhält der Spieler für jedes besetze Feld einen weiteren Würfel dazu.

**Steuerung**
Durch Linksklick kann man den Würfelstapel auf einem eigenen Feld selektieren und durch einen weiteren Klick auf ein Nachbarfeld verschieben.
Die Kamera wird durch Mausbewegungen bei gedrückter rechter Maustaste gesteuert.
Der Kamerazoom befindet sich auf +/-.
ESC beendet das Spiel.
F2 aktiviert/deaktiviert die FPS & Debugausgabe
F3 wechselt zischen Wireframe und Fill Mode
F7 wechselt zwischen Display Lists und Immediate Mode (im VBO Modus nicht verfügbar)
F8 aktiviert/deaktiviert Frustum Culling

**Effekte**
*Per Pixel Lighting*

Pixelbasiertes Beleuchtungsmodell, implementiert in GLSL. *[1]*



Im Fragment Shader wird aus 2 OpenGL Lichtquellen
gl_LightSource[0] ... Point
gl_LightSource[1] ... Spot
ein Phong Beleuchtungsmodell berechnet.

*Shadow Mapping*

Für beide Lichtquellen wird eine Depthmap in ein Framebuffer Object gerendert. Der Test ob ein Fragment beleuchtet wird oder nicht erfolgt mittels shadow2DProj im Fragment Shader. *[2]*

**Nichttriviale Objekte**
Jedes im Spiel befindliche Objekt (mit Ausnahme der Skybox) ist ein texturiertes Collada Modell. Das Holzbrett, die Würfel und die Feldkacheln wurden mit Blender erstellt und texturiert.
Die Ente stammt aus den Collada Beispieldateien.

**Animierte Objekte**
Würfel werden beim Verschieben programmatisch animiert.

**Beschleunigung der Sichtbarkeitsberechnung**
Um die Darstellung zu beschleunigen, werden nicht sichtbare Objekte nicht gerendert. Zum Testen der Sichtbarkeit wird eine Bounding Box gegen das View Frustum der Kamera getestet. Die Bounding Box wird auch zum Ermitteln des selektierten Würfelstapels verwendet. (Culling: Box-Frustum Test, Picking: Box-Ray Intersection *[3]*)

**Experimentieren mit OpenGL**
Das Spiel unterstützt drei Modi zum Rendern der Geometriedaten. Standardmäßig werden Positions-, Normalen- und Texturdaten in Vertex Buffer Objekte geladen und mittels glVertexPointer gerendert. Alternativ kann man das Spiel auch mit aktiviertem Immediate Mode starten und dann zwischen Immediate Mode und Display Lists wechseln.
Ein Wechsel zwischen VBO Rendering und den anderen beiden Modi im laufenden Spiel ist nicht möglich, da sich die Geometriedaten im VBO Modus ausschliesslich im Speicher der Grafikkarte befinden. (siehe beiliegende *.bat Files)

**Implementierungsdetails**
Modelloader: FCollada
Window System Abstraktion: GLUT
Bildformat der Texturdaten: PNG (libpng)
Extension Library: GLEW

**Quellen**

[1] "Point Light Per Pixel"
http://www.lighthouse3d.com/opengl/glsl/index.php?pointlight

[2] "Shadow Mapping Tutorial Using GLSL"
http://www.cs.cmu.edu/afs/cs/academic/class/15462/web.06s/asst/project3/shadowmap/

[3]Williams "An Efficient and Robust Ray-Box Intersection Algorithm", 2005
http://jgt.akpeters.com/papers/WilliamsEtAl05/