



INTO THE MOLEHILL  
DER BLICK INS LOCH

# Über mich

- **Malte Beyer**
- **Teamleiter Entwicklung bei der Powerflasher GmbH, Aachen**
- **Leitung und Programmierung komplexer Kundenprojekte**
- **Schulungen, Vorlesungen**

# Über Powerflasher

- Gegründet 1997
- Carlo Blatz CEO/Founder
- Über 40 feste Mitarbeiter
- Aachen, Hamburg, San Francisco, Linz(A)
- Spezialisiert auf multimediale On- und Offline-Lösungen
- Gestartet als Flash-Agentur, heute viel mehr



# Ziel des Vortrags

- **Grundlagen** von Molehill erklären
- Praktischen **Einstieg** in die Entwicklung mit Molehill vermitteln
- Konfiguration der **Entwicklungsumgebung**
- **Einsatzgebiete** von Molehill vorstellen

# Agenda

1. Was ist Molehill?
2. Konfiguration der Entwicklungsumgebung
3. Entwicklung auf low-level Ebene
4. Entwicklung mit 3D Frameworks
5. Molehill im 2D Einsatz
6. Ausblick

**1. WAS IST MOLEHILL?**

# Molehill Grundlagen

- „Molehill“ ist der Codename für eine **low-level 3D API** von Adobe für den Flash Player und AIR
- Ermöglicht die Verwendung der **GPU** und somit hardwareseitige Beschleunigung
- Dadurch ist „echtes“ **3D in Echtzeit** auf diversen Zielsystemen möglich
- Erstes final Release frühestens Mitte 2011

# Was macht die GPU?

- **GPU** = Graphics Processing Unit  
Grafikprozessor (Hardware)
- Übernimmt **rechenintensive Berechnungen** der Computergrafik und entlastet die CPU
- Ermöglicht enorme **Performance-Gewinne**, da ein Großteil der Rechenzeit von der CPU auf die GPU verlagert wird (auch für Notebooks, Netbooks, mobile Endgeräte)

# Vergleich

- Ohne GPU können bislang **~2.000** Polygone flüssig gerendert werden
- Mit GPU können **~250.000 Polygone** flüssig gerendert werden (**~125 mal mehr**)



# Hierarchie

- Content, der von der GPU gerendert wird, liegt immer **hinter der DisplayList**
- Molehill Objekte haben nie etwas mit der DisplayList zu tun, sondern **liegen in einem eigenen Layer**

# Hierarchie

- **DisplayList**



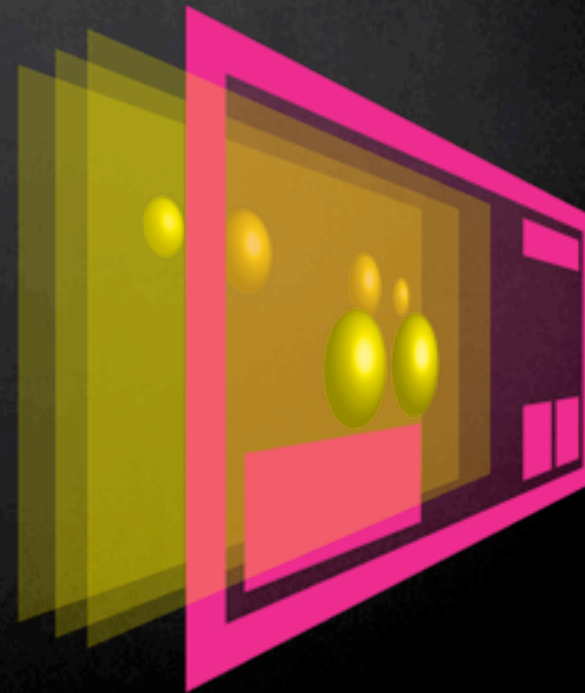
## Hierarchie

Molehill und StageVideo sind losgelöst von der DisplayList

# Hierarchie

- **DisplayList**
- **Molehill**

Stage3D[0]  
Stage3D[1]  
Stage3D[n]



## Hierarchie

Molehill und StageVideo sind losgelöst von der DisplayList

# Hierarchie

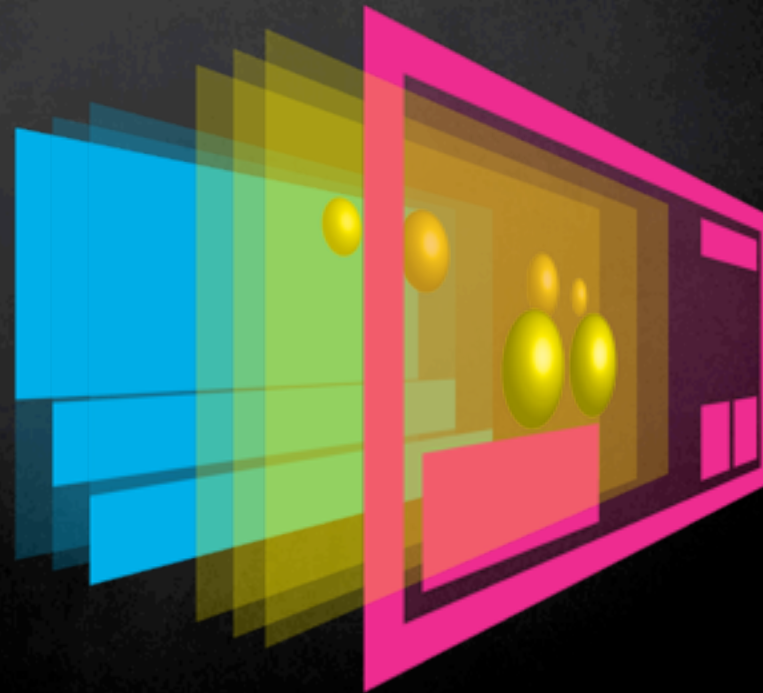
- **DisplayList**

- **Molehill**

Stage3D[0]  
Stage3D[1]  
Stage3D[n]

- **StageVideo**

StageVideo[0]  
StageVideo[1]  
StageVideo[n]



## Hierarchie

Molehill und StageVideo sind losgelöst von der DisplayList

# Wird Molehill immer unterstützt?

- Automatisch **DirectX9** unter Windows, **OpenGL 1.3** unter MacOS und Linux, **OpenGL ES 2.0** auf mobilen Endgeräten
- Automatischer **Fallback auf CPU** mittels des neuen Rasterizers „**SwiftShader**“ (ca. 10 mal schneller als der alte)

# Zielgruppe von Molehill

- **Erfahrene 3D Entwickler**, die gerne low-level programmieren
- **Entwickler von 3D Bibliotheken**
- **Alle Flash Entwickler**, die die 3D Frameworks nutzen möchten



## 2. KONFIGURATION DER ENTWICKLUNGSUMGEBUNG

# Einmalige Konfiguration

- Beschrieben unter <http://blog.powerflasher.de/macht-euer-fdt-molehill-ready>
- Compiler Argument „-swf-version=13“ hinzufügen
- In der einbettenden HTML Seite `params.wmode="direct";` hinzufügen

# Einmalige Konfiguration

- Flash Player **Incubator Build** installieren
- **Flex SDK** herunterladen und patchen
- Flex SDK zum **FDT** hinzufügen
- FDT **Project Template** installieren  
(liefert Compiler Settings, HTML-Seite, wmode, Launcher, M2D SWC)

# Fazit der Konfiguration

- Ein mal **konfigurieren**
- Danach einfach neues Flash Projekt mit Typ „**Web**“ | „**AS3 Molehill**“
- Alles **bereit** zum Loslegen

### 3. ENTWICKLUNG AUF LOW-LEVEL EBENE

# Funktionsweise

- **Vertices** definieren die Form unseres 3D Objekts
- **Shader** modifizieren die Position der Vertices und das Erscheinungsbild der Texturen
- Shader werden auf die Grafikkarte (**GPU**) hochgeladen und dort berechnet



# Shader

- Shader für Molehill werden in **AGAL** (Adobe Graphics Assembly Language) geschrieben
- AGAL kann selber in **Maschinensprache** geschrieben werden (schwierig)
- Oder man nutzt **Pixel Bender 3D** zur Erzeugung von AGAL

# Beispiel

- Beispiel von Mikko unter <http://labs.jam3.ca/2011/03/molehill-getting-started/>

# Fazit

- Implementierung **nicht einfach**
- **Viel Code** für wenig Output
- Maximale **Kontrolle**, da nicht von anderen Frameworks abhängig

## 4. ENTWICKLUNG MIT 3D FRAMEWORKS

# Welche Libs nutzen Molehill?

- Alternativa3D
- Away3d
- CopperCube
- Flare3D
- Minko
- Sophie3D
- Yogurt3D
- ...

# Am Beispiel Away3d

- **Sourcen mit FDT auschecken von**  
**<http://away3d.googlecode.com/svn/trunk/broomstick/Away3D>**



# FDT Project Reference

- Neues Feature mit **FDT 4.3**
- Referenzierung eines anderen Projekts mit **allen Features** (Navigation, Refactoring, ...)
- Vorteil: immer auf dem neuesten Stand durch **einfaches syncen**

# FDT Project Reference anlegen

- Rechtsklick auf das away3D Projekt → Flash Project → **Add Flash Nature** (AS3, Flex 4.0, Player Version 10.0)
- Rechtsklick auf SRC Folder → Classpath → **Add to classpath**
- Rechtsklick auf Projekt → Properties → **FDT Build Path** → Projects → Add Projects → Away3D → OK

# Beispiele

- Spinning Cube
- OBJ Import

# Sinnvolle Einsatzgebiete

- 3D Echtzeit Games
- 3D Konfiguratoren (Auto Konfiguratoren, Tassen bedrucken, Raumplaner, ...)
- 360 Grad Drehungen
- Spezialeffekte (Rauch, Feuerwerk, ...)
- Alles, was 3D Echtzeit erfordert

# Nicht sinnvolle Einsatzgebiete

- Wenn vorhandenes 2.5D auch reicht und die Performance ausreichend ist
- Wenn viele Elemente über der DisplayList liegen müssen
- Wenn keine Echtzeit gebraucht wird  
→ 3D Videos könnten auch reichen

## Fazit (nur bezogen auf Away3d)

- **Einfache** Implementierung
- Momentan werden noch **nicht alle Features** unterstützt
- Noch **fehlerhaft**
- Nur eine **Frage der Zeit**, bis sowohl Molehill, als auch die 3D Frameworks fehlerfrei laufen



## 5. MOLEHILL IM 2D EINSATZ

## 2D

- Molehill macht manchmal **auch für 2D** Sinn
- **GPU** anstatt DisplayList
- Enorme **Performance**-Gewinne
- **Einschränkungen** durch Entkopplung von der DisplayList

# M2D Framework

- 2D Molehill Framework von Ely Greenfield  
<https://github.com/egreenfield/M2D>
- Open Source
- Viele Beispiele von Tom Krcha unter  
<http://flashrealtime.com/demos/m2d/>  
[M2DSamples.zip](#)

# Einschränkungen M2D

- Bislang **keine MovieClips** (stattdessen SpriteSheets)
- Bislang **keine MouseEvents**
- Bislang **nicht dokumentiert**
- Bislang **wenig Funktionalität**

→ nur eine Frage der Zeit (noch **im Aufbau**)

# Beispiele

- SpriteSheet

SpriteSheets können aus einer SWF mit „Zoë“ von Grant Skinner erzeugt werden:

<http://easeljs.com/zoe.html>

# Sinnvolle Einsatzgebiete

- Partikel
- 2D Games
- Spezialeffekte (Rauch, Feuerwerk, ...)
- Überall dort, wo es sehr viele DisplayObjects gibt
- Überall dort, wo 2D Objekte sehr viel CPU erfordern



# Nicht sinnvolle Einsatzgebiete

- Wenn **User Interaktion** mit der Maus benötigt wird (noch)
- Wenn Probleme mit der **Tiefen-Sortierung** zwischen GPU und DisplayList bestehen
- Für **Standard-Implementierungen**, die kaum CPU erfordern

# Fazit

- Momentan **noch nicht** richtig nutzbar (nur für Experimente)
- In Zukunft könnte es **sehr interessant** werden
- Wahrscheinlich zukünftig auch **weitere 2D Frameworks**
- Der Einsatz sollte gut überlegt werden, da es **Einschränkungen** gibt

## 6. AUSBLICK

# Was wird Molehill bringen?

- Molehill wird mit Release **fehlerfrei** sein
- Alle wichtigen **3D Frameworks** werden Molehill mit allen Features und **stabil** unterstützen
- Es wird **mächtige 2D Frameworks** geben
- Flash wird ein großer Konkurrent im Netz, wenn es um **3D Echtzeit** geht (Flash Player vs. Unity Plugin)

DOWNLOAD

# Download dieser Folien & Beispiele

- Alle eben gezeigten **Folien und Beispiele** können unter folgender URL heruntergeladen werden:

**<http://blog.powerflasher.de/molehill-vortrag-auf-der-ffk11-beyond-tellerrand>**



DANKE

# POWERFLASHER GMBH BELVEDEREALLEE 5 52070 AACHEN

[www.powerflasher.de](http://www.powerflasher.de)

---

**MALTE BEYER**

**Teamleiter Entwicklung**

**E** [mb@powerflasher.de](mailto:mb@powerflasher.de)

**T** +49 241 91880 230