

# Kill Bird: Volume 1

## Dokumentation zur 3. Abgabe

CG 2/3 Übungen, SS 2006

Charwot Raphael, 534, 0426010, e0426010@student.tuwien.ac.at

Hausmair Robert, 532, 9709998, e9709998@student.tuwien.ac.at

19.06.2006

### Inhaltsverzeichnis

<b>1</b>	<b>Hintergrund</b>	<b>2</b>
<b>2</b>	<b>Kurzbeschreibung</b>	<b>2</b>
<b>3</b>	<b>Das Spiel</b>	<b>2</b>
3.1	Bau-Modus . . . . .	2
3.2	Kampf-Modus . . . . .	3
<b>4</b>	<b>Techniken</b>	<b>3</b>
4.1	Laden von Modellen . . . . .	3
4.2	Instanziierung von Modellen und Texturen . . . . .	3
4.3	Quad-Tree-Szenengraph . . . . .	3
4.4	Hierarchisches Frustum-Culling . . . . .	3
4.5	SmartPointers und Memory-Watcher . . . . .	3
4.6	Dynamische Terrainerstellung . . . . .	3
4.7	Sound-System und eigene Musik . . . . .	4
4.8	GUI-Manager . . . . .	4
4.9	Physik . . . . .	4
4.10	Sky-Box . . . . .	4
4.11	Grafik-Effekte . . . . .	4
4.12	Pfeile bleiben stecken . . . . .	6
4.13	Angriffswellen . . . . .	7
4.14	Effekt-Switches . . . . .	7
4.15	Sonstiges . . . . .	7
<b>5</b>	<b>Libraries</b>	<b>7</b>

## 1 Hintergrund

Die geflügelte Bevölkerung dieses Planeten hat entgültig genug von Legebatterien, Grillhuhn-Imbissbuden, Daunnen-Betten und Vogelgrippe-paranoiden Massenschlachtungen. Daher beschließen die Vögel, sich gegen die Menschheit zu wehren und fangen an, die ersten kleinen Dörfer zu attackieren. Du bist verantwortlich für die Sicherheit eines dieser ersten Angriffsziele und musst es um jeden Preis verteidigen, um eine weitere Invasion dieser bösartigen Vogelarmee zu verhindern.

## 2 Kurzbeschreibung

Das Spiel besteht wie geplant aus zwei Modi:

1. Bau-Modus
2. Kampf-Modus

Nach dem Start des Spiels erscheint das Startmenü. Nach einem Klick auf "New Game" wird das erste Level geladen und man beginnt im Bau-Modus. Nach einer gewissen Zeit, die rechts oben angezeigt wird, beginnt die Vogel Invasion und es wird in den Kampfmodus gewechselt. Mit Alt+F4 oder ESC kann das Spiel jederzeit beendet werden.

## 3 Das Spiel

### 3.1 Bau-Modus

Im Baumodus wird eine spezielle Kamera-Steuerung verwendet, die das horizontale Scrollen über der Karte erlaubt, wie es in Strategiespielen üblich ist.

#### 3.1.1 Wirtschaft

Es gibt verschieden Gebäude zur Auswahl, die unterschiedlich viel kosten. Für jedes überlebende Gebäude erhält man am Ende der Kampf-Phase einen gewissen Anteil seiner Baukosten als Punkte und Geld gutgeschrieben. Dieser Anteil hängt vom Zustand des Gebäudes ab: je beschädigter es ist, desto weniger Geld und Punkte ist es wert. Mit Ausnahme der Hauptstatue werden die Gebäude zwischen den Levels nicht repariert, passt also besser gut auf sie auf.

#### 3.1.2 Verteidigung

Der Turm ist ein Spezialgebäude, das zwar keine Einnahmen einbringt, aber dafür selbst auf herannahende Vögel schießt und so bei der Verteidigung des Dorfs mithilft. Richtig platziert sind diese Türme eine wirkungsvolle Waffe gegen die gefiederten Invasoren.

## 3.2 Kampf-Modus

Hier kann sich der Spieler mit der üblichen WSAD+MAUS-Steuerung in der Landschaft bewegen, mit SPACE kann man springen und mit LEFT SHIFT laufen. Die Vögel kommen in mehreren Wellen auf das Dorf zu und versuchen, es zu zerstören. Der Kampf ist gewonnen, wenn alle Vögel zerstört wurden. Das Spiel ist verloren, wenn die wunderbare Hauptstatue zerstört wurde.

## 4 Techniken

Unter anderem werden folgende Techniken in unserem Spiel eingesetzt:

### 4.1 Laden von Modellen

Dank des FBXSDKs können Modelle im FBX- oder OBJ-Format importiert und in unsere eignen Modell-Strukturen konvertiert werden.

### 4.2 Instanziierung von Modellen und Texturen

Modelle werden nur einmal geladen und können dann beliebig oft in der Szene dargestellt werden. Hierbei werden nur die nötigen Daten (Position, Richtung, Skalierung) dupliziert, die eigentlichen Modelldaten sind nur einmal im Speicher.

Gleiches gilt für das Laden und Darstellen von Texturen. Fordern mehrere Modelle die gleiche Textur, so wird diese nur einmal geladen und allen nötigen Modellen zugeteilt.

### 4.3 Quad-Tree-Szenengraph

Alle Modell-Instanzen werden mittels eines Quad-Tree-Szenengraphen hierarchisch unterteilt und verwaltet.

### 4.4 Hierarchisches Frustum-Culling

Der Szenengraph und alle darin enthaltenen Modell-Instanzen werden mittels des Kamera-Frustums auf Sichtbarkeit überprüft, wobei zuerst Sphere-Tests und im Zweifelsfall zusätzliche Bounding-Box-Tests durchgeführt werden.

### 4.5 SmartPointers und Memory-Watcher

Viele der Klassen und Objekte verwenden SmartPointer, um reference counting und die rechtzeitige Zerstörung der Objekte zu automatisieren. Dies und der Einsatz eines effektiven Memory-Watchers verhindern das Auftreten von memory und ressource leaks.

### 4.6 Dynamische Terrainerstellung

Mittels einer Heightmap wird das Terrain zur Laufzeit erstellt und in "QuadTree-gerechte" Stücke zerlegt, die dann in den Szenengraphen eingehängt werden können.

Das Terrain wird mit einer Color- und einer Detail-Textur versehen und mittels einer "Vegetation"-Map mit verschiedenen Bäumen bepflanzt.

## 4.7 Sound-System und eigene Musik

Ein einfaches Sound-System wurde mittels FMOD erstellt sowie eigens für das Spiel komponierte Hintergrund-Musik eingebaut.

## 4.8 GUI-Manager

Es wurde ein vollwertiger GUI-Manager implementiert, der sowohl hierarchisches Input-Messaging, dynamische Menüerstellung als auch das Anzeigen von MODALEN Fenstern erlaubt.

## 4.9 Physik

Das AGEIA PHYSX-SDK wurde genutzt, um Collision Detection und simple Physik zu implementieren. Leider hat dieses SDK in seiner aktuellen FREE Version (die kommerzielle ist nicht betroffen) einen Bug, wegen dem Character Controller (= die Vögel), die sich in der selben Gruppe befinden immer kollidieren, auch wenn man diese Kollisionen explizit deaktiviert. Dieser Bug wurde uns bereits Anfang Juni von einem Ageia-Mitarbeiter bestätigt und ein Hotfix versprochen, welcher jedoch bis zur Abgabe noch nicht verfügbar war.

## 4.10 Sky-Box

Eine hübsche Skybox zielt den Himmel. Sie wird im Camera space mit deaktiviertem Z-Buffer gerendert (als allererstes Objekt) und dreht sich mit dem Spieler mit. Durch das Rendern im Camera space muss sie nicht verschoben oder skaliert werden und ist immer um die aktuelle Ansicht zentriert.

## 4.11 Grafik-Effekte

Folgende Effekte wurden gemäß der Anforderungen für die 3. Abgabe implementiert:

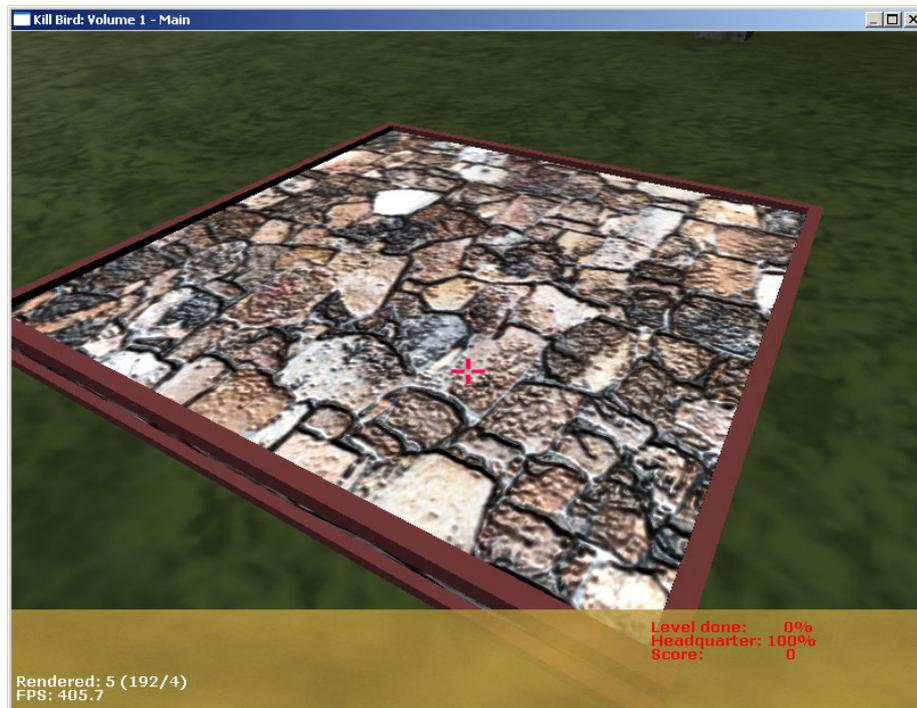
### 4.11.1 Environment mapping (3 Punkte)

In einigen Gebäudefenstern und an der Hauptstatue sieht man, dass sich der Himmel an diesen Oberflächen reflektiert. Implementierung mittels SPHERICAL REFLECTION MAPPING (mit Standard OpenGL-Befehlen und einer 2. Textur-Einheit).



#### 4.11.2 Bump mapping (3 Punkte)

Einige Gebäudeflächen sind mit bump mapping versehen, wodurch sie besonders plastisch und realistisch aussehen (zB das Podest auf dem der Spieler startet). Hierfür wurden in CG passende Vertex- und Fragment-Shader geschrieben, welche sogar spekulare Highlights zeigen. (Einige Ideen wurden aus [Math3D] genommen.)



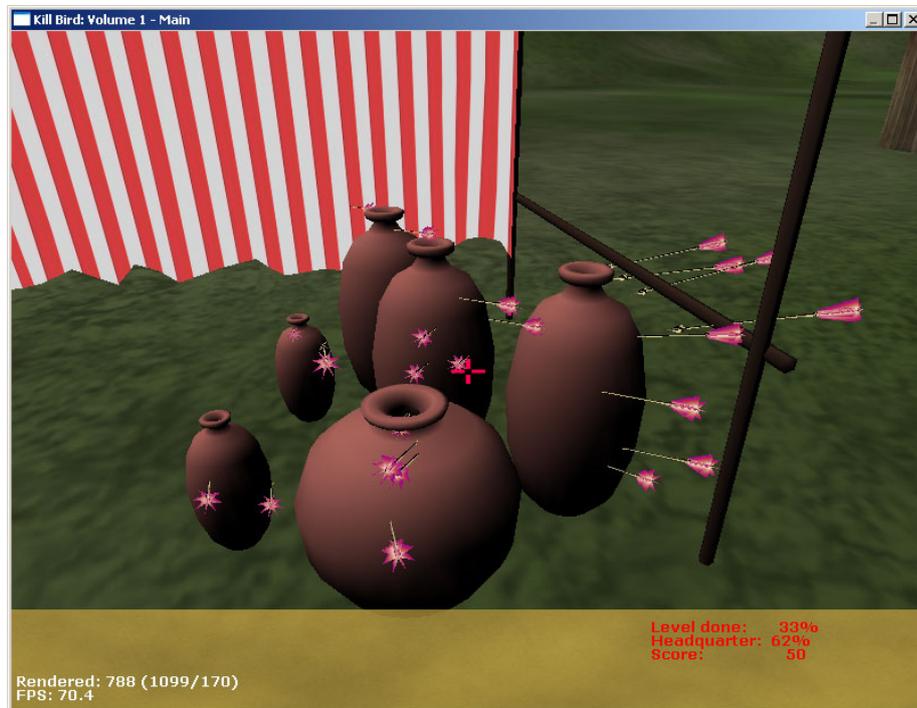
#### 4.11.3 Transparenz (2 Punkte)

Einige GUI-Elemente (Fadenkreuz, Battle-Fenster) sowie die Bäume und der Steinboden werden (teilweise) transparent dargestellt. Dafür werden diese Elemente sortiert und nach den undurchsichtigen Objekten gerendert.

Insgesamt wurden also 8 Punkte von unserem 2-Mann-Team implementiert.

#### 4.12 Pfeile bleiben stecken

Bemerkt ein Pfeil eine Kollision mit einem anderen Objekt, so heftet er sich an dieses und bleibt einige Zeit stecken und bewegt sich mit, falls sich das Objekt weiter bewegt (zB bei Vögeln). Um möglichst realistische Trefferpunkte zu erhalten wird nach einer gemeldeten Kollision eine eigene Ray-Intersection auf Polygon-Level mit dem getroffenen Objekt ausgeführt, um den nächsten Kontaktpunkt am Objekt herauszufinden. Somit ist es möglich, die Pfeile genau auf ein getroffenes Objekt zu "kleben", obwohl die Collision-Volumes wesentlich größer aufgelöst sind. (Diese sind im Allgemeinen nur bounding boxes.)



#### 4.13 Angriffswellen

Für jedes Level können per Skript Angriffspunkte definiert werden, an denen dann nach einer gewissen Spielzeit im Kampfmodus eine bestimmte Anzahl von Vögeln erstellt wird. Diese SPAWN-POINTS erlauben eine einfache und flexible Levelerstellung mit herannahenden Vogelschwärmen aus verschiedenen Richtungen.

#### 4.14 Effekt-Switches

Wie gefordert, können mit den Funktionstasten die einzelnen Effekte ein- und ausgeschaltet werden. Im Log-Fenster wird dabei jeweils der neue Zustand angezeigt.

#### 4.15 Sonstiges

Zusätzlich zu den oben genannten Punkten wurden zahlreiche Features implementiert, die uns bei der Entwicklung des Spieles unterstützt haben. Dazu zählen unter anderem ein hierarchisches Input-System, eine Implementierung des State-Patterns für die Spiellogik, die Trennung von Spielobjekt-Logik und -Darstellung, ein fähiges Logging-System uvm.

## 5 Libraries

Folgende Libraries wurden bisher verwendet:

**Boost-Library** Zum Parsen der settings.ini-Datei, in der alle wichtigen Spiel-Einstellungen liegen.

**fmod** Zum Laden und Abspielen der Sounds

**FBXSDK** Zum Laden von Modellen im FBX- und OBJ-Format

**PhysX-SDK** Zur Implementierung der Physik. Siehe Bug-Bericht unter 4.9.

**Glew** Um OpenGL-Extensions einfach zu nutzen.

**FluidStudios' MemMgr** Zur Erkennung von memory leaks.

**cg** Die Shadersprache cg wurde für das bump mapping verwendet.

## Literatur

[Math3D] Mathematics for 3D Game Programming and Computer Graphics,  
2nd Edition