D o k u m e n t a t i o n

ShootMii

-

Flashgame mit Wiimote-Steuerung

Christian Langenmair
Klinkerberg 9
86152 Augsburg
Tel.: +49 160 96643613
christian.langenmair@fh-augsburg.de

Matrikel-Nr. 913067

Matthias Nüßl
Rauwoffstr. 5
86152 Augsburg
Tel.: +49 176 24674990
m.nuessl@i-forge.de

Matrikel-Nr. 913072

Studienarbeit im Fach Interaktion
SS 2010
Hochschule Augsburg
Interaktive Medien – 6. Semester

Betreuer: Thomas Rist

Abstract:

In dieser Studienarbeit wird untersucht, ob die Steuerung eines in Flash umgesetzten Spieles mit zwei Wiimotes sinnvoll umgesetzt werden kann. Die Arbeit beschreibt die Herangehensweise an diese Fragestellung mitsamt einigen Implementierungsdetails und kommt zu dem Ergebnis, dass die Wiimote-Steuerung einen erheblichen Mehrwert für den Spieler mit sich bringt. Allerdings wird auch deutlich, dass einige Probleme mit der Bluetooth-Verbindung und daher auch mit dem Auslesen der Wiimote-Parameter unter Verwendung der aktuellen Version des verwendeten WiiFlash-Servers nur schwer vermeidbar sind.

Categories and Subject Descriptors:

H.5.2 [User Interfaces]

- Input devices and strategies (Nintendo Wiimote)
- Haptic I/O

General Terms:

Human Factors, Measurement, Reliability

Keywords:

Flash, ActionScript, WiiFlash, Wiimote, Bluetooth

Inhaltsverzeichnis

1. Motivation 4

2. Verwandte Arbeiten 4

3. Anwendungsszenario und Use Cases 5

3.1. Anwendungsszenario 5

3.2. Definition der Use Cases 5

4. Entwurf eines Prototyps 7

4.1. Die WiiFlash-Technologie 7

4.2. Konzeptioneller Aufbau des Prototypen 7

5. Implementierung 9

5.1. Hard- und Softwareumgebung 9

5.1.1. Hardware 9

5.1.2. Software 9

5.2. Kern der Implementierung 10

5.3. Umfang des Prototypen 12

6. Evaluation 12

6.1. Ziele der Evaluation und Methodik 12

6.2. Durchführung der Evaluation 13

6.2.1. Mehrwert und Spassfaktor 13

6.2.2. Technische Umsetzbarkeit und Zuverlässigkeit 13

6.3 Interpretation der Evaluationsergebnisse 14

7. Zusammenfassung 14

7.1. Erzielte Ergebnisse 14

7.2. Erweiterungsmöglichkeiten / Ausblick 15

8. Referenzen 15

1. Motivation

Auf der einen Seite ist die von Adobe entwickelte Flash-Technik das am weitesten verbreitete Format für interaktive Anwendungen im Internet. Durch den hohen Grad an Portabilität und Plattformunabhängigkeit eignet es sich hervorragend dazu, Spiele und andere Programme zu verbreiten und auf jedem Computer ausführen zu können.
Auf der anderen Seite ist die Wii von Nintendo eine der erfolgreichsten Spielkonsolen der Welt. Zwar ist die Wii-Plattform leistungstechnisch ihren Konkurrenten deutlich unterlegen, kann aber durch die Steuerung mit den Wiimotes punkten, die sich mit ihren Bewegungs- und Infrarotsensoren von den bekannten Steuergeräten wie Maus, Tastatur und Gamecontroller grundlegend unterscheidet.
Ziel dieser Arbeit ist nun zu untersuchen, ob sich diese beiden Techniken verbinden lassen und deren Vorteile sinnvoll kombiniert werden können. Hierzu soll ein auf Flash basierendes Spiel entwickelt werden, das sich mit zwei Wiimotes steuern lässt.

2. Verwandte Arbeiten

Die zentrale Technik, um Wiimotes mit Flash-Applikationen zu verbinden ist der WiiFlash-Server, der von Joa Ebert und Thibault Imbert entwickelt wurde. Mit WiiFlash wurden bereits einige Projekte umgesetzt. Einige davon sollen an dieser Stelle erwähnt werden.

Das Projekt „Twiidee“[i] wurde von Fabien Bizot entwickelt und ist als Lernprogramm für Anatomie-Vorlesungen gedacht. Hier können mit Hilfe einer Wiimote dreidimensional modellierte Organe bewegt werden. Mit Gesten können die Studenten zum Beispiel die Ansicht eines Herzens verändern und skalieren. Da die 3D-Modelle sehr detailliert sind, wird hier mit 3D-PDFs in Adobe Air gearbeitet, um die im Adobe Reader zum Einsatz kommende Render-Engine und somit OpenGL nutzen zu können.[ii]

Ein weiteres, mit WiiFlash entwickeltes Projekt ist Stella Artois "Virtual Tap" [iii]. Die belgische Biermarke Stella Artois veranstaltet jährlich einen Wettbewerb, bei dem die Teilnehmer versuchen, das Bier auf perfekte Art und Weise zu zapfen. Da das dabei verbrauchte Bier dem Hersteller zu teuer wurde, ließ er eine WiiFlash-Applikation entwickeln, bei der mit Wiimote und Nunchuck die komplette Prozedur des Bierzapfens simuliert werden kann. Die Entwickler wurden für dieses Projekt mit Bronze beim Cannes Lions Media Wettbewerb ausgezeichnet.

Darüber hinaus gibt es bereits einige weitere Projekte, die auf Flash und Wiimote basieren. Etwa diverse musikalische Installationen, DJ-Programme und eine Graffiti-Spray-Simulation. Die Wii-Fernbedienung lässt sich multifunktional einsetzen, entwickelt wurde sie jedoch für einfaches, intuitives Spielen. Und genau in diesem Bereich gibt es bisher wenig vorzeigbare Projekte.

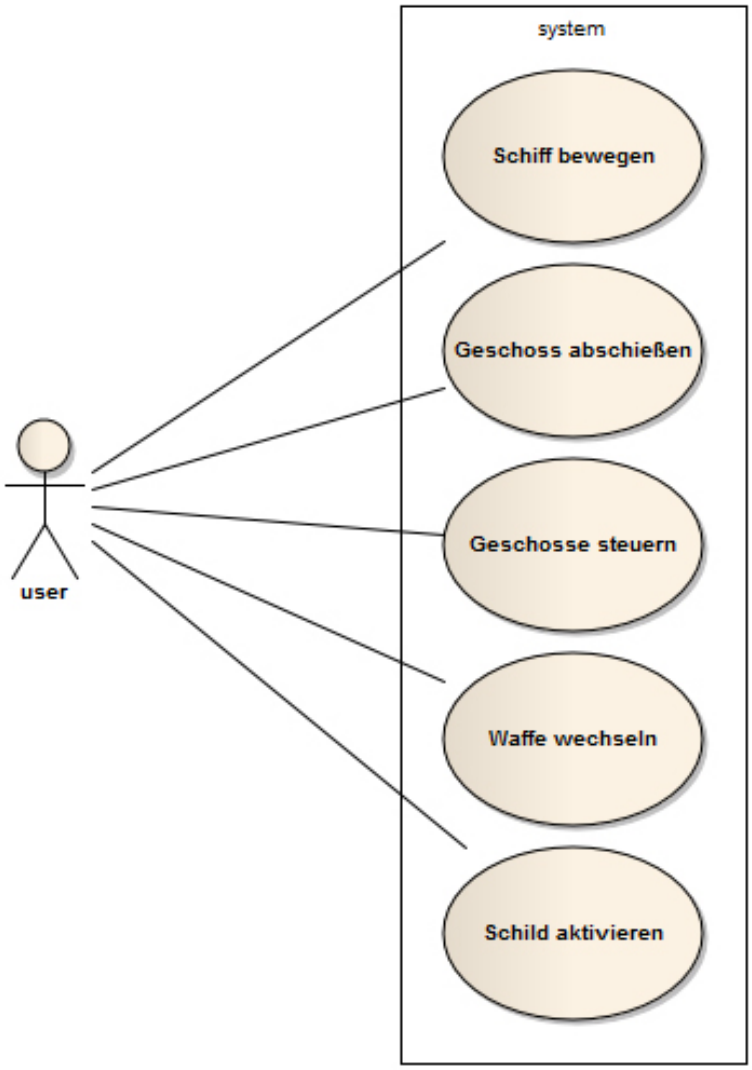
Somit hebt sich das in dieser Arbeit vorgestellte Projekt von den anderen Applikationen vor allem dadurch ab, dass hier der eigentliche Sinn und Zweck von Wiimotes wieder aufgegriffen wird, nämlich das Spielen. Auf der Wii-Konsole besitzen die meisten Spiele einen Multiplayer-Modus, was gerade bei einfacheren Spielideen den Spassfaktor enorm steigert. Daher wurde auch unser Projekt für zwei Wiimotes, also auch für zwei Spieler ausgelegt.

3. Anwendungsszenario und Use Cases

3.1. Anwendungsszenario

Entwicklung eines Flashgames mit Wiimote-Steuerung. Es spielen zwei Spieler gegeneinander auf einem zweidimensionalen, rechteckigen Spielfeld, wobei sich jeder Spieler nur auf seiner eigenen Hälfte bewegen kann. Der Spieler kann Kugeln auf seinen Gegner abschiessen, deren Bewegung er durch das Neigen der Wiimote steuern kann. Mit dem Steuerkreuz auf der Wiimote kann er den gegnerischen Kugeln ausweichen. Außerdem gibt es verschiedene PowerUps, die zufällig auf dem Spielfeld auftauchen und entweder die Bewegungsgeschwindigkeit des Spielers erhöhen oder sein Schild wieder aufladen.

3.2. Definition der Use Cases



Use Case Name:	**Schiff bewegen**
Scenario:	
Der User bewegt sein Schiff indem er die gewünschte Richtung auf dem Steuerkreuz der Wiimote drückt. Es ist auch möglich, sich diagonal zu bewegen, indem man z.B. oben und rechts gleichzeitig gedrückt hält.	
Constraints:	
Pre-condition	Der User kann sich nur auf seiner Seite des Spielfeldes bewegen. Das Ziel seiner Bewegung muss also auf seiner Seite liegen.
Post-condition	Das Schiff des Users befindet sich auf der neuen Position, auf jeden Fall aber innerhalb seiner Hälfte.

Use Case Name:	**Geschoss abschießen**
Scenario:	
Der User kann ein Geschoss abschießen, indem ein den „2“-Button auf der Wiimote drückt	
Constraints:	
Pre-condition	Das Geschoss ist aufgeladen
Post-condition	Das Geschoss wurde abgeschossen und kann durch Neigung gesteuert

Use Case Name:	**Geschoss steuern**
Scenario:	
Der User kann ein Geschoss steuern, indem der die Wiimote neigt	
Constraints:	
Pre-condition	Mindestens ein Geschoss befindet sich im Flug
Post-condition	Das Geschoss fliegt in der Richtung weiter, in die die Wiimote geneigt wurde.

Use Case Name:	**Waffe wechseln**
Scenario:	
Der User wechselt zur nächsten Waffe, indem er den Button „1“ auf der Wiimote drückt	
Constraints:	
Pre-condition	
Post-condition	Das neue Geschoss wurde ausgewählt

Use Case Name:	**Schild aktivieren**
Scenario:	
Der User kann sein Schild aktivieren, indem er die Taste „A“ auf der Wiimote drückt. Während das Schild aktiviert ist, kann das Schiff nicht getroffen werden. Das Schild wird deaktiviert, wenn „A“ losgelassen wird.	
Constraints:	
Pre-condition	Das Schild muss noch Energie haben
Post-condition	Es wird je nach Länge des Tastendrucks Energie abgezogen.

4. Entwurf eines Prototyps

4.1. Die WiiFlash-Technologie

Das hier beschriebene Spiel wird mit Wiimotes gesteuert. Eine der Besonderheiten einer Wiimote ist die Möglichkeit über drei Lagesensoren die momentane Ausrichtung auszulesen. Diese Ausrichtung kann über drei Achsen (x, y und z) und deren Kombinationen bestimmt werden. Zusätzlich verfügt die Wiimote über Infrarotsensoren, mit denen auch die Position bezüglich eines Fixpunktes (Sensorleiste) gemessen werden kann. Die Infrarottechnik wird allerdings in diesem Projekt nicht angewandt, von Interesse ist lediglich der Wert der y-Achse und der Zustand verschiedener Knöpfe.
Die Verbindung zwischen Wiimote und PC findet über Bluetooth statt, wobei der Bluetooth-Socket des Betriebssystems die angemeldeten Wiimotes dem WiiFlash Server zur Verfügung stellt. Dieser wiederum kann bereits sämtliche Sensoren der Wiimote auslesen und gibt diese Werte dann an die Flashplayer-Sandbox weiter. Die mit WiiFlash mitgelieferte WiiFlash-API stellt alle benötigten Funktionen und Klassen für die Wiimote-Ansteuerung aus ActionScript 3 zur Verfügung.

4.2. Konzeptioneller Aufbau des Prototypen

Der Prototyp wurde in der Programmiersprache ActionScript 3 geschrieben und ist gemäß dem MVC-Pattern aufgebaut. Alle View-Komponenten wurden in Adobe Flash gestaltet und befinden sich in der Bibliothek der Datei ShootMii.fla. Zusätzlich hat jede View-Komponente ein entsprechendes gleichnamiges Gegenstück in AS3, wodurch sich zusätzliche Eigenschaften für diese Komponenten implementieren lassen. Im View-Package liegen also alle sichtbaren Elemente des Spiels, wie zum Beispiel die Repräsentation der Spieler (Ship.as), der Geschosse (Ball.as) und das Menü (Menu.as), welches alle Spieldaten anzeigt.
Im Package Model liegen diejenigen Klassen, in welchen sämtliche Spieldaten gespeichert sind. Da im Prototyp keine Persistenzschicht implementiert ist, stehen im Model anfangs immer die initialen Daten. So wird zum Beispiel in der Klasse PlayerModel.as die Anzahl der Leben und die Geschwindigkeit des Ships festgelegt, sowie die verfügbaren Waffen und PowerUps erstellt. Während des Spiels werden alle Veränderungen direkt in den Model-Klassen gespeichert, damit die Controller-Klassen immer mit dem aktuellen Werten arbeiten können (z.B. veränderte Geschwindigkeit).
Im Package Controller befindet sich die gesamte Spiellogik, also alle Klassen, die den Status der Applikation verändern. Die Hauptklasse ApplicationController.as dient dazu, alle anderen Controller zu instanziieren und stellt außerdem einige Methoden zur Verfügung (z.B. das Wechseln der Spielerwaffen). Desweiteren liegen im selben Package die Controller zum Steuern der Ships und der Geschosse (ShipMotionController.as und BallMotionController.as), sowie die Steuerklassen für die HitTests (ShipHitController.as und BallHitController.as). Die Klassen BallTimer.as und BallTimerController.as regeln die Aufladezeit der Geschosse und PowerUpController.as schließlich steuert das zeitlich zufällige Auftauchen neuer PowerUp-Komponenten.
Die wichtigste Controller-Klasse ist allerdings WiiMoteController.as, welche die beiden Wiimotes anfangs initialisiert und anschließend alle benötigten Event-Listener vergibt. Die Events, auf die die Listener hören, werden von der Klasse org.wiiflash.events.ButtonEvent definiert, welche Bestandteil der WiiFlash-API ist. Events werden lediglich für die Knöpfe auf der Wiimote gebraucht.

Das Auslesen der y-Achse geschieht indirekt über einen ENTER_FRAME - EventListener, der in der Klasse BallMotionController aufgerufen wird, um die Bewegungs-Richtung der Geschosse zu steuern. Außerdem wird die Drehung des Ships in indirekte Relation zur y-Achse der Wiimote gesetzt, indem in der Klasse ShipMotionController ebenfalls die Methode zum Auslesen des y-Sensors aufgerufen wird.



5. Implementierung

5.1. Hard- und Softwareumgebung

Die erste Testumgebung für die Kommunikation zwischen Wiimote und Flash wurde auf einem MacBook Pro unter Mac OS X 10.6.1 realisiert. Hier ergaben sich massive Probleme mit dem Verbinden der Wiimotes unter dem OS X-internen Bluetooth-Socket. Zum einem brach die Verbindung nach einigen Sekunden ab, zum Anderen konnte die in Java geschriebene Variante des WiiFlash-Servers die Wiimotes nicht korrekt auslesen. Nach langer Recherche und Fehlersuche wurde schließlich beschlossen, das Projekt mit dem folgenden Hard- und Softwaresetup zu realisieren:

5.1.1. Hardware

- MacBook Pro (Late 2007) mit Intel Core 2 Duo und 4GB RAM
- 2x Nintendo Wiimote

5.1.2. Software

Betriebssystem:

- Mac OS X 10.6.1
- VMware Fusion 3.0.0
- Windows 7 Professional als Virtuelle Maschine unter VMware Fusion

Entwicklungsumgebung:

- Adobe FlashBuilder 4
- Adobe Flash CS5

Verwendete Bibliotheken:

- WiiFlash-API für ActionScript 3

Programmiersprache:

- ActionScript 3

Anwendungsumgebung:

- WiiFlash-Server 0.4.5. (.NET – Version)
- Adobe Flash Player 10

5.2. Kern der Implementierung

Das Hauptanliegen dieser Studienarbeit besteht darin, eine funktionsfähige Ansteuerung von zwei Wiimotes aus einer Flash-Applikation zu implementieren. Daher soll hier als Kern der Implementierung die Klasse WiiMoteController.as genauer beschrieben werden.

Zunächst werden in der Klasse WiiMoteController die Methoden initWiiMote1() und initWiiMote2() aufgerufen, in denen die Wiimotes initialisiert und verbunden werden und anschließend die benötigten EventListener vergeben werden.

Auszug aus der Methode initWiiMote1():

```
private function initWiimote1():void
{
	wiiMote1 = new Wiimote();
	wiiMote1.connect();

	wiiMote1.addEventListener(ButtonEvent.DOWN_PRESS, w1ButtonPressed);
	wiiMote1.addEventListener(ButtonEvent.UP_PRESS, w1ButtonPressed);
	wiiMote1.addEventListener(ButtonEvent.RIGHT_PRESS, w1ButtonPressed);
	wiiMote1.addEventListener(ButtonEvent.LEFT_PRESS, w1ButtonPressed);

	wiiMote1.addEventListener(ButtonEvent.DOWN_RELEASE, w1ButtonReleased);
	wiiMote1.addEventListener(ButtonEvent.UP_RELEASE, w1ButtonReleased);
	wiiMote1.addEventListener(ButtonEvent.RIGHT_RELEASE, w1ButtonReleased);
	wiiMote1.addEventListener(ButtonEvent.LEFT_RELEASE, w1ButtonReleased);
	( … )
}
```

Die Bewegungen der beiden Ships werden in der Klasse ShipMotionController in jedem Frame berechnet (ENTER_FRAME – EventListener), wobei sich die Richtung nach vier Boolean-Variablen richtet, die in den Methoden w1ButtonPressed(e:ButtonEvent), bzw. w2ButtonPressed(e:ButtonEvent) und in w1ButtonReleased(e:ButtonEvent), bzw. w2ButtonReleased(e:ButtonEvent) gesetzt werden.

Auszug aus der Methode w1ButtonPressed():

```
private function w1ButtonPressed(e:ButtonEvent):void
{
	switch(e.type)
	{
		case 'downPress':
			appController.smc1.keyRight = true;
			break;
		case 'upPress':
			appController.smc1.keyLeft = true;
			break;
		case 'leftPress':
			appController.smc1.keyDown = true;
			break;
		( … )
	}
}
```

In den eben beschriebenen Methoden wurden abhängig vom Zustand der Wiimote-Buttons (pressed oder released) bestimmte Aktionen ausgeführt.

Für das Auslesen der Neigungswerte sind allerdings EventListener nicht sinnvoll, da die Bluetooth-Technik und somit auch die Wiimotes mit einer Frequenz von 100 Hz senden und EventListener somit 100 Mal in der Sekunde ausgelöst werden würden. Die Flash-Applikation ist allerdings für eine Wiederholrate von 25 Frames pro Sekunde ausgelegt und genau sooft sollen auch die Werte für die y-Achse ausgelesen werden.
Aus diesem Grund wurde zur Aktualisierung der Achsenwerte die Methode getAngle() implementiert, die von den Klassen BallMotionController und ShipMotionController jeweils von einem ENTER_FRAME – EventListener (also 25 Mal pro Sekunde) aufgerufen wird.

Auszug aus der Methode getAngle():

```
public function getAngle(p_player:int):Number
{
	if(p_player == 1)
	{
		if(_angle1.toString() != "NaN")
		{
			_angle1 = lowPassFilter(_angle1, wiiMote1.sensorY);
		}
		else
		{
			_angle1 = wiiMote1.sensorY;
		}
		return _angle1;
	}
	( … )
}
```

Durch den Aufruf wiiMote1.sensorY wird der aktuelle Wert des y-Achsen-Sensors ausgelesen, jedoch ist es nicht sinnvoll, diesen Wert direkt an die MotionController weiterzugeben, da sich dieser auch bei nur kleinen Bewegungen der Wiimote trotzdem sprunghaft ändert und somit ein unannehmbares Zittern der entsprechenden visuellen Komponenten zur Folge hätte. Dieser Effekt würde noch verstärkt, da in den MotionController-Klassen der Neigungswert (der stets zwischen -1 und 1 liegt) bei der Umrechnung in das Winkelmaß mit einem Faktor >1 multipliziert wird. Daher wurde ein LowPassFilter integriert, der dieses „Zittern“ unterbindet.

Die Methode lowPassFilter()mit Definition der Konstante FILTER_FACTOR:

```
private const FILTER_FACTOR:Number = 0.2;

private function lowPassFilter( oldValue:Number, value:Number ):Number
{
	return  value * FILTER_FACTOR + oldValue * (1.0 - FILTER_FACTOR );
}
```

Im LowPassFilter wird der aktuelle Wert des y-Sensors mit dem Wert verrechnet, der im vorhergegangenen Aufruf der Methode getAngle() ausgelesen wurde. Durch die Kumulation mit der Konstante FILTER_FACTOR (Wert: 0,2) ist der von getAngle() zurückgegebene Wert nur zu 20% vom aktuellen y-Sensor-Wert abhängig, aber zu 80% vom alten Wert, was bei kleinen und bei großen Bewegungen der Wiimote immer eine weiche und flüssige Animation ermöglicht.

5.3. Umfang des Prototypen

Das Hauptziel dieser Projektarbeit, also die Steuerung eines Flashgames mittels zweier Wiimotes wurde im Prototypen vollständig implementiert. Die Spieler haben die Möglichkeit, ihr Ship mit dem Steuerkreuz zu bewegen, Kugeln auf den Gegner abzuschießen und diese im Flug durch Neigung der Wiimote zu steuern, die Schusswaffe zu wechseln und ein Schutzschild zu aktivieren. Einige offizielle Wii-Anwendungen benutzen darüber hinaus Gestenerkennung. Diese wurde aus zwei Gründen im Prototyp nicht umgesetzt:

1. Die in der WiiFlash-API mitgelieferten Klassen, die als Grundlage für die Gestenerkennung dienen könnten (z.B. HistoryPeakDetection.as) stellen nur sehr rudimentäre Funktionen zur Verfügung. Schon das Auslesen der Peaks, also der Höhepunkte der Bewegungen, funktioniert nur sehr eingeschränkt und wenig zuverlässig, was eine weitergehende Interpretation der Daten sehr schwierig macht.
2. Obwohl die Gestenerkennung ein sehr interessantes Gebiet, vor allem für Wiimotes ist, wurde im hier beschriebenen Prototypen darauf verzichtet. Das entwickelte Flashgame würde durch die Gestenerkennung keinen Mehrwert haben, da sich der Spieler durch die Kombination aus Tastendruck-Befehlen und Neigungssteuerung bereits auf zwei grundverschiedene Steuerungsmechanismen gleichzeitig konzentrieren muss. Kämen hier noch weitere Spielfunktionen (z.B. durch Schütteln der Wiimote) hinzu, wäre die Steuerung schlichtweg überladen. Da im Allgemeinen die Steuerung für ein Spiel möglichst einfach gehalten werden soll, haben wir uns gegen die Gestensteuerung entschieden.

Abseits der Gestensteuerung beinhaltet der Prototyp alle Funktionen, die ein lauffähiges Multiplayer-Spiel benötigt. Die eigentliche Aufgabenstellung wurde sogar um einige Aspekte der Spiellogik erweitert, wie zum Beispiel die zufällig auftauchenden PowerUps, das Schutzschild und mehrere Schusswaffen. Außerdem wird im Prototyp bei einigen Spielzuständen haptisches Feedback über die Rumble-Funktion der Wiimotes gegeben.

6. Evaluation

6.1. Ziele der Evaluation und Methodik

In der Evaluation des vorliegenden Projektes soll vor allem geprüft werden, ob eine Wiimote-Steuerung für Flash-Applikationen in der Praxis sinnvoll einsetzbar ist. Die Praxistauglichkeit einer solchen Anwendung hängt vor allem von zwei Faktoren ab:

- Mehrwert und erhöhter Spassfaktor für den Spieler
- Technische Umsetzbarkeit und Zuverlässigkeit

Zur Evaluation des ersten Faktors durften einige Testpersonen mit dem Prototyp spielen. Da Multiplayer-Funktionalität implementiert wurde, spielten immer zwei Personen an einem Computer gegeneinander. Anschließend wurden die Testpersonen bezüglich der Intuitivität und des Spassfaktors befragt.
Der zweite wichtige Faktor wurde überprüft, indem der Prototyp auf insgesamt drei verschiedenen Testumgebungen getestet wurde. Hierzu wurde die Applikation mitsamt WiiFlash-Server auf einem MacBook Pro unter OS X Snow Leopard und auf einem über VMware Fusion emulierten Windows 7 (32 Bit) installiert. Als zusätzliche Testumgebung diente ein Windows 7 (64 Bit) auf einem Desktop-PC mit Bluetooth-Dongle.

6.2. Durchführung der Evaluation

6.2.1. Mehrwert und Spassfaktor

Sämtliche Personen, mit denen die Evaluation durchgeführt wurde, besaßen sowohl normale bis fortgeschrittene PC-Kenntnisse, als auch einige Erfahrungen an der Wii-Konsole. Die Steuerung mit der Wiimote war für alle Beteiligten daher sehr intuitiv. Auch die Übertragung der Bewegungen auf die Spiel-GUI war flüssig und nachvollziehbar.
Durch die einigermaßen schnelle Spielgeschwindigkeit waren häufig recht unkontrollierte und übertriebene Bewegungen zu beobachten, die Testspieler fuchtelten häufig wie wild mit der Wiimote herum, was zwar zum einen zu einer sehr ungenauen Steuerung führte, zum anderen aber den Spielspass deutlich steigerte.
Ergebnis der anschließenden Befragungen war, dass sich die Wiimote sehr wohl für Flashgames eignet. Die Übertragung der Wii-Steuerung auf ein PC-Spiel war eine willkommene Abwechslung zu der ansonsten stets gleichen Steuerung mit Tastatur und Maus.

6.2.2. Technische Umsetzbarkeit und Zuverlässigkeit

Für die Tests der technischen Aspekte wurde die Komplexität, Dauer und Fehleranfälligkeit für folgende Aktionen überprüft:

1. Herstellung der Verbindung zwischen Wiimotes und systemseitigem Bluetooth-socket
2. Start des WiiFlash-Servers und Registrierung der Wiimotes
3. Auslesen der Wiimote-Parameter aus ActionScript 3

Bereits beim ersten Punkt unterschieden sich die drei getesteten Systeme deutlich: Unter OS X Snow Leopard konnte die Verbindung zwar ohne Probleme hergestellt werden, allerdings kam es reproduzierbar zu großen Problemen beim Anmelden einer zweiten Wiimote. Teilweise wurde hier die erste Wiimote sofort wieder abgemeldet, teilweise waren beide Wiimotes angemeldet, konnten aber vom WiiFlash-Server nicht ausgewertet werden. Auf dem Desktop-PC konnte (vermutlich aufgrund des minderwertigen Bluetooth-Dongles) überhaupt keine Verbindung zu einer Wiimote hergestellt werden. Lediglich unter dem emulierten Windows 7 auf dem MacBook konnte die Verbindung sogar mit vier Wiimotes gleichzeitig zuverlässig hergestellt werden.
Der weitere Test des WiiFlash-Servers beschränkte sich daher auf das letztgenannte System. Die .NET Variante des Servers lief stets zuverlässig und die relevanten Daten der Wiimotes wurde immer korrekt an die Flashplayer-Sandbox übergeben, allerdings nur unter der Voraussetzung, dass bei der Initialisierung stets die richtige Reihenfolge eingehalten wurde:
Zuerst muss die Bluetooth-Verbindung hergestellt werden, erst danach darf der WiiFlash-Server gestartet werden. Nachdem dieser die Wiimotes registriert hat, kann erst die eigentliche Applikation gestartet werden.

6.3 Interpretation der Evaluationsergebnisse

Die Evaluation des Prototypen brachte auf der Steuerungsebene ausschließlich positive Ergebnisse. Der Spassfaktor beim Spielen wurde durch den Einsatz der Wiimotes gravierend gesteigert, auch die Intuitivität der Steuerung wurde von den Testpersonen sehr geschätzt. Neben der Bewegungssteuerung brachten die Wiimotes zusätzlich den Vorteil, dass das Flashgame mit kabellosen Controllern gesteuert wurde, die keine Unterlage benötigen. Man kann sich beim Spielen im Gegensatz zu den meisten anderen Computerspielen also gemütlich zurücklehnen, was vor allem beim Spielen zu zweit große Platz- und Bequemlichkeitsvorteile bringt.

Auf der technischen Ebene wurden allerdings einige Probleme festgestellt. Zwar konnte die Applikation zuverlässig gestartet werden, allerdings nur mit einigem Aufwand. Die Unerlässlichkeit der genauen Reihenfolge beim Anmelden der Wiimotes führt zu der Erkenntnis, dass ein solches Projekt nur in einer administrierten Installation sinnvoll genutzt werden kann.

7. Zusammenfassung

7.1. Erzielte Ergebnisse

In der vorliegenden Studienarbeit wurde durch Implementierung eines Prototyps überprüft, ob sich die Steuerung mit Wiimotes grundsätzlich für Flash-Applikationen eignet. Anfangs wurden bereits existierende, technisch ähnliche Projekte recherchiert, beschrieben und in Bezug zu dem hier angewandten Anwendungsszenario gebracht. Anschließend wurde die WiiFlash-Technik vorgestellt und der Aufbau des Prototyps beschrieben.
Die Implementierung der Applikation in ActionScript mit Hilfe der WiiFlash-API gestaltete sich einfach und zielführend. Allerdings kam es mit zwei der drei Testumgebungen zu Problemen mit der Bluetooth-Verbindung. Dies führte zu der Erkenntnis, dass die Kombination aus Flash und Wiimote zwar auf einer korrekt administrierten Testumgebung einwandfrei funktioniert, ein Programm dieser Art einem öffentlichen Nutzerkreis zugänglich zu machen, macht aber wegen der Komplexität des Initialisierungsvorgangs wenig Sinn.
Auch bei der Evaluation des Projekts wurde festgestellt, das sich Wiimotes zur Steuerung von Computerspielen, also auch von Flash-Spielen prinzipiell sehr gut eignet. Alle Vorteile der Wii-Konsole lassen sich hierauf anwenden und dem Spieler wird so ermöglicht, die Bewegungssteuerung für interaktive Applikationen ohne großen finanziellen Aufwand zu erleben.

Sollte es in Bezug auf die Einrichtung der Bluetooth-Verbindung und des WiiFlash-Servers noch Fortschritte in der Verbesserung der Benutzerfreundlichkeit und Zuverlässigkeit geben, ist diese Technologie eine interessante Alternative zu „traditionellen“ Steuerungsmethoden für Computerspiele.

7.2. Erweiterungsmöglichkeiten / Ausblick

Bezüglich der Interaktion besteht die interessanteste Erweiterung des Projekts in der Implementierung der Gestenerkennung. Die setzt allerdings zum Einen eine bessere Erkennung der Peaks in den Klassen der WiiFlash-API voraus, zum Anderen sollte einer solchen Funktion auch eine Berechtigung im Hinblick auf den Mehrwert für den Spieler zugrunde liegen.

Die sinnvollste Erweiterung der WiiFlash-Technik im Allgemeinen wäre eine Optimierung der Initialisierungsprozedur. Eventuell ist hier eine Installationsroutine denkbar, die einen eigenen Bluetooth-socket mitsamt den benötigten Scripts für die Anmeldung der Wiimotes mitbringt. So könnte dann dem Spieler nach dem Starten der Flash-Applikation mit einfachen Anweisungen die benötigten Schritte erklärt und teilweise abgenommen werden. Auch die Möglichkeit, im laufenden Spiel Wiimotes neu anzumelden (z.B. nach Verbindungsabbruch) könnte so realisiert werden.

8. Referenzen

i) http://www.lafabrick.com/blog/2009/03/02/1079-fitc09-twiidee-wiiflash-and-rock-n-roll/
ii) http://wiiflash.bytearray.org/
iii) http://directdaily.blogspot.com/2009/06/stella-artois-virtual-tap.html

i http://www.lafabrick.com/blog/2009/03/02/1079-fitc09-twiidee-wiiflash-and-rock-n-roll/
ii http://wiiflash.bytearray.org/
iii http://directdaily.blogspot.com/2009/06/stella-artois-virtual-tap.html