

Optische Bewegungserkennung mithilfe einer Wii Remote

**Wahlpflichtfach 44 - Bildverarbeitung und
Algorithmen
Wintersemester 2008/2009**

Sebastian Janzen

1. März 2009



Inhaltsverzeichnis

1	Was ist eine WiiRemote?	4
2	Technische Daten der WiiRemote	5
2.1	Die Hardware	5
2.2	Das Protokoll	5
2.3	Probleme	6
3	Technische Daten der Infrarotstifte	7
3.1	Verwendete IR-Dioden	7
3.2	Aufbau und Gehäuse	7
4	Software	9
4.1	Verwendete Bibliotheken	9
4.1.1	BlueCove	9
4.1.2	WiiRemoteJ	10
4.2	Programm WRTest	10
4.3	Programm WiimoteWhiteboard	12
4.3.1	Besonderheit BlueCove	12
4.3.2	Bedienung WiimoteWhiteboard	12
5	Beobachtungen	16
5.1	Empfindlichkeit der IR-Kamera	16
5.2	Leistung der IR-LEDs	16
5.3	Praxistauglichkeit	16
6	Fazit	17
6.1	WiiRemote als SmartBoard	17
6.2	Ist WiimoteWhiteboard praxistauglich?	17
6.3	Die Zukunft der Computereingabe	17

Abbildungsverzeichnis

1	Fertiger Infrarot-Stift	7
2	Schema des Infrarot-Stifts	8
3	<i>BlueCove</i> Softwareschichten	9
4	Screenshot WRTest im <i>EXTENDED</i> Modus	10
5	Screenshot WRTest im <i>FULL</i> Modus	11
6	WiimoteWhiteboard Hauptfenster	12
7	WiimoteWhiteboard IR Kamera Monitor	13
8	WiimoteWhiteboard Kalibrierungsdialog	14
9	Projektionsprinzip	14
10	WiimoteWhiteboard Kalibrierungsdetails	15



Tabellenverzeichnis

1	IR-Kamera Antwortframe für das Detaillevel „Short“	5
2	IR-Kamera Antwortframe für das Detaillevel „Long“	6

Listings

1	Programm WRTest	20
---	---------------------------	----



Dieses Dokument behandelt die Verwendung einer Wii Remote in Kombination mit einer Infrarotlichtquelle in Form eines Stifts zur Ermittlung der Stiftposition auf einer Projektionsfläche. Ziel ist die Umsetzung der Stiftbewegung in Mausbewegungen.

1 Was ist eine WiiRemote?

[Wik09b] Die WiiRemote, oder verkürzt WiiMote, ist ein Eingabegerät für die Spielkonsole *Wii* von Nintendo. Der Begriff setzt sich aus dem Namen der Konsole und dem englischen Begriff *Remote* (Fernbedienung) zusammen. Beim Verkaufsstart im Jahre 2006 hob sich die Spielkonsole von ihrer Konkurrenz durch die ungewöhnliche Bedienung ab. Anders als bei konkurrierenden Produkten wurde hier zusätzlich zu den bewährten Knöpfen und Steuerkreuzen auf dem Eingabegerät mehrere zusätzliche Sensoren verbaut, insbesondere drei Beschleunigungssensoren und eine **Infrarotkamera**.

Ziel war es, mit den Informationen über Beschleunigung und der Position des Eingabegeräts im Raum eine wesentlich intuitivere Bedienmöglichkeit zu schaffen. Dies erschloss einen neuen Markt für Spielehersteller, welcher bis heute der Konsole großen Erfolg sichert.

Mit diesem Erfolg wuchs auch das Interesse an anderweitiger Verwendungsmöglichkeit der WiiMote. Was anfangs nur auf die Spielkonsole selber abzielte [Tea09] wurde später auf die WiiMote selber spezialisiert - teils mit großem Erfolg. Das Protokoll, mit dem Daten von und zur Wiimote geschickt werden wurde zu großen Teilen entschlüsselt. [Wii09b] Viele Projekte stützen sich auf diese Arbeit, sodass man heute auf fertige Libraries zurückgreifen kann, die teils Open Source, teilweise auch kostenlos zur Verfügung stehen.

Ein in diesen Kreisen sehr bekannter Name lautet Johnny Chung Lee, der sich mit innovativen und interessanten Projekten einen Namen in der Szene gemacht hat. Seine Arbeiten begannen als Studienprojekte für die School of Computer Science [Wii09a] in Pittsburgh, Pennsylvania. Die Idee, mit Infrarotlicht emittierenden Stiften und einer Wiimote ein „Berührungsempfindliches“ Whiteboard zu konstruieren, stammt von ihm. Die Quelltexte in der Sprache C# und ein Video in dem die Funktionsweise demonstriert wird, stellt er auf seiner Wiimote Projektseite unter der GPL zur Verfügung.

Ziel dieser Ausarbeitung war es, die Ergebnisse von Johnny Chung Lee nachzuvollziehen und Probleme bei der Realisierbarkeit aufzudecken, da eine detaillierte Abhandlung dazu nicht verfügbar ist.





2 Technische Daten der WiiRemote

2.1 Die Hardware

Die Wiimote ist mit folgenden Sensoren ausgestattet:

- Einem Beschleunigungssensor [Dev09]
- Einem Lautsprecher
- Zwölf Tastern
- Einer 1024 × 768 XGA Infrarotkamera [Wik09a]

Der Hersteller des IR-Sensors, PixArt, veröffentlichte leider kein Datenblatt. Innerhalb des Sichtfeldes der Wiimote erfasst der intern über einen I²C-Bus angesprochene Sensor [Jü09] in einem Sichtfeld von 45° Infrarotlichtquellen. Das Filterglas vor der Optik des Sensors schirmt ihn vor einem Teil anderer Wellenlängen als $\approx 940\text{nm}$ Licht ab. [Wik09a] Aus diesem Grund verwenden die IR-Stifte in diesem Projekt entsprechende LEDs mit 940nm . (Siehe auch Kapitel 3.1)

Die Verarbeitungsgeschwindigkeit innerhalb des Sensors beträgt 100 Hz, über Bluetooth lassen sich die Sensordaten mit 50 Hz abfragen. Der Sensor wird von einem Quartz in der Wiimote getaktet, Experimente mit anderen Taktgebern haben gezeigt, dass sich der Sensor sehr flexibel, also auch mit anderen Geschwindigkeiten ansteuern lässt. [Jü09]

2.2 Das Protokoll

Bei der Initialisierung der Wiimote werden verschiedene Konfigurationsparameter gesetzt, aufgrund derer sich dann die abrufbaren Antwortframes zusammensetzen. Dies ist nötig, da aufgrund der durch Bluetooth eingeschränkten Übertragungsrate nicht alle Sensorinformationen gleichzeitig übertragbar sind.

Wird beispielsweise der Beschleunigungssensor aktiviert und sollen gleichzeitig Daten der Kamera übertragen werden, so können letztere nicht in voller Detailgenauigkeit erfasst werden.

Insgesamt gibt es elf Frames, die abgerufen werden können. Das Kameraframe beinhaltet die Positionen alle zum Detaillevel gehörenden erfassten Punkte. Existiert nur ein Punkt, so sind die anderen Felder mit $0xFF$ gefüllt.

BIT	7	6	5	4	3	2	1	0
0					X0			
1					Y0			
2		Y0		X0		Y1		X1
3					X1			
4					Y1			
5					X2			
6					Y2			
7		Y2		X2		Y3		X3
8					X3			
9					Y3			

Tabelle 1: IR-Kamera Antwortframe für das Detaillevel „Short“

Die in 1 dargestellten Felder enthalten zwei Byte pro Koordinate zuzüglich zwei Bit in den Reihen zwei und sieben. Insgesamt stehen im Short-Modus also zehn Bit pro Punkte und Achse zur Verfügung, was auf der



X-Achse dezimal genau 1023 Zuständen entspricht. Das bedeutet, das in diesem Modus die Auflösung der Kamera exakt abgedeckt ist.

BIT	7	6	5	4	3	2	1	0	
0					X				
1					Y				
2	Y				X			S	
3	0				XMIN				
4	0				YMIN				
5	0				XMAX				
6	0				YMAX				
7	0	0	0	0	0	0	0	0	
8									
									INTENSITY

Tabelle 2: IR-Kamera Antwortframe für das Detaillevel „Long“

In Tabelle 2 ist zu sehen, dass hier ein Punkt verfolgt werden kann. Da dieses Frame jedoch zweimal pro Antwort vorkommt, liegt die Genauigkeit bei 16 Bit pro Achse. Das sind mit dezimal 65536 Zuständen das 64 fache von 1024. Das entspricht einer achtfachen Subpixelgenauigkeit auf beiden Achsen.

$$2^{16} = 65536 = 64 \cdot 1024$$

$$\sqrt{65536} = 8$$

In diesem Modus wird jedoch nur ein Punkt verwertet. Zusätzlich sind in sieben Bit Auflösung die Bounding Box als MAX- und MINX sowie MAX- und MINY angegeben. Außerdem wird die Intensität mit acht Bit angegeben.

2.3 Probleme

In der Praxis konnte ich mit der Java-Bibliothek *WiiRemoteJ* (Siehe Kapitel 4.1.2) die gegebenen Werte zwar abfragen, jedoch schienen sie nur einmal mit Werten befüllt worden zu sein. Danach änderte sich deren Wert nicht mehr. Auch wurde überprüft, ob es sich um eine aufspannende Bounding Box aus mehreren IR-Punkten handelte. Fehlanzeige.



3 Technische Daten der Infrarotstifte

3.1 Verwendete IR-Dioden

Die verwendeten IR-LEDs besitzen eine Wellenlänge von 940nm . [Kin09] Sie werden mit ca. $\approx 50\text{mA}$ betrieben. Mit speziellen LEDs können auch höhere Leistungen und damit auch Erkennungsleistungen erzielt werden. Johnny Chung Lee verwendet eine in Deutschland schwierig zu bekommende IR-LED von Vishay. [Vis09] Probeexemplare lassen sich unter <http://www.vishay.com> bestellen.¹ Optimieren ließe sich der Stift durch eine geregelte Stromquelle. Der Einfachheit und Handhabung wegen wurde jedoch eine einfache AAAA Batterie verwendet. NiMH Akkus können laut LED-Spezifikation auch verwendet werden, dies wurde jedoch nicht ausprobiert.



Abbildung 1: Fertiger Infrarot-Stift

3.2 Aufbau und Gehäuse

Ein vorhandener Taschenlampenstift wurde durch den Wechsel der LED und einsetzen einer kleinen Lochrasterplatte mit Schalter und Kontroll-LED zum Infrarotstift. Das Schema ist in Abbildung 2 zu sehen. Weitere Abbildungen können unter [Jana] eingesehen werden.

Durch den Schalter an der Rückseite des Stifts wird der Stromfluss zwischen dem Minuspol der Batterie durch die Klemmfeder und der Gehäuseaußenwand des Stifts hergestellt. Die Lochrasterplatte ist mit der Gehäusewand und dem Pluspol der Batterie verbunden. Die Kontroll-LED zeigt nun durch Leuchten an, das der Stift betriebsbereit ist. Der Taster in Form eines SMD-Kurzhubtasters schließt bei Betätigung den Stromkreis zwischen IR-LED und Batterie. Der dabei entstehende Spannungsabfall reicht aus, um die nötige Betriebsspannung für die Kontroll-LED zu unterschreiten. Die Kontroll-LED zeigt also den negierten Betriebszustand der IR-LED an. Ist die IR-LED defekt, leuchtet die Kontroll-LED dauerhaft, da kein Spannungsabfall zustande kommt. Leuchtet die Kontroll-LED nicht auf, ist entweder die Batterie leer, der Taster defekt oder die IR-LED kurzgeschlossen. Um letzteres Risiko zu minimieren, sind relevante Stellen mit Silikonschlauch ummantelt worden, wie auf Abbildung 1 zu sehen ist.

¹Für diesen Vortrag wurden zehn TSAL6400 bestellt, die seit dem 28. Februar jedoch erst vorliegen.

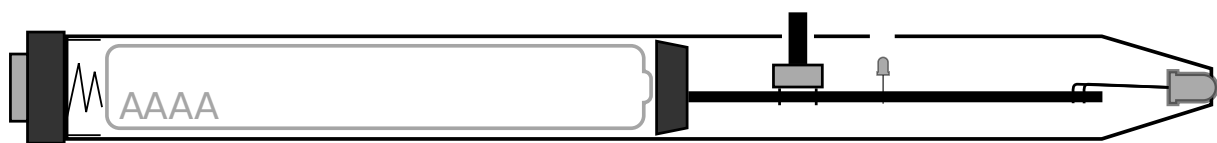


Abbildung 2: Schema des Infrarot-Stifts



4 Software

4.1 Verwendete Bibliotheken

4.1.1 BlueCove

Diese Bibliothek verwendet ein eigenes Java Native Interface, welches in C++ geschrieben ist, um direkt auf den Bluetooth Stack des Betriebssystems zuzugreifen. Dies geschieht weitestgehend Betriebssystemunabhängig. Den Ursprung findet das Projekt bei Intel, welche das Projekt jedoch aufgab und es unter GPL veröffentlichte. Seit Version 2.1 ist es unter der Apache Software License in Version 2.0 veröffentlicht. Der GPL-Teil wurde ausgelagert in eine eigene Bibliothek, die nötig ist, um auf den Linux Bluetooth Stack *BlueZ*, welcher selbst unter GPL steht, zuzugreifen. In diesem Fall wird *BlueCove* und zusätzlich *BlueCove GPL* geladen, welches die zusätzlichen Klassen und den JNI-Part bereitstellt. [Blu09]

Für das Projekt wurde Version 2.1 inklusive der GPL-Erweiterung verwendet.

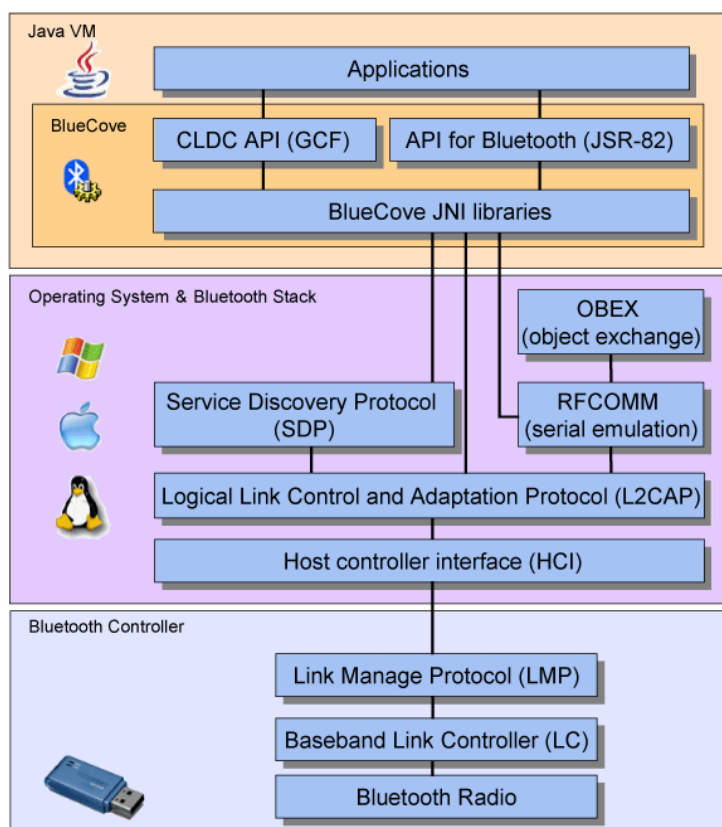


Abbildung 3: BlueCove Softwarearchitektur

Die im Projekt verwendete Software setzt auf der *Applications*-Schicht auf. Beim initialisieren von *BlueCove* durch eine Applikation wird die JSR-82 API geladen und die nötige *BlueCove JNI library* geladen. Diese greift wiederum auf den betriebssystemeigenen Bluetooth-Stack zu. Unter Windows wäre dies zum Beispiel der *Widcomm*-Stack, der mit sehr vielen Bluetooth USB-Sticks mitgliedert wird. Unter Linux ist dies in der Regel *BlueZ*. Dieses greift wiederum über das *Host Controller Interface*, also über USB, auf die Bluetooth-Hardware zu. Bluetooth-Stacks bieten verschiedene Dienste an, darunter den *L2CAP*-Dienst, auf den die meisten anderen aufbauen. Er erlaubt uns auf sehr hardwarenaher Schicht auf Bluetooth-Geräte zuzugreifen. Alternativ, jedoch mit Einschränkungen, kann der *RFCOMM*-Dienst verwendet werden. Dieser simuliert einen COM-Port



auf dem Computer, an den virtuell das Bluetooth-Gerät, das über Funk verbunden ist, angeschlossen ist. Dieser Weg verzögert jedoch die Kommunikation durch den Simulationsaufwand. [EDEK09]

4.1.2 WiiRemoteJ

WiiRemoteJ ist eine spärlich dokumentierte Bibliothek, die auf *BlueCove* aufbaut. Sie lässt sich unter [Wii09c] herunterladen. Es existiert keine Projektwebseite, der Author diskutiert in diversen Foren mit. Die JavaDoc reicht jedoch aus, um funktionierende Applikationen auf die Beine zu stellen. Für das Projekt wurde die Bibliothek in der Version 1.5 verwendet.

Die Bibliothek setzt auf dem L2CAP-Protokoll auf, um möglichst verzögerungsfrei auf die *WiiRemote* zuzugreifen.

Prinzipiell wird ein Objekt der Klasse *WiiRemoteJ* erzeugt, die Methode *findRemotes()* aufgerufen und ein Listener an die gefundene Wiimote gebunden. Der Listener ist das Interface *WiiRemote* implementierendes oder von *WiiRemoteAdapter* erbenendes Objekt, welches - auszugsweise - folgende Methoden implementiert:

boolean isConnected() Gibt zurück, ob eine Wiimote verbunden ist.

boolean isIRSensorEnabled() Gibt an, ob die Kamera aktiviert ist.

void setIRSensorEnabled(boolean enabled, int mode, IRSensitivitySettings sensitivitySettings) Aktiviert die Kamera mit den übergebenen Einstellungen.

Der Parameter *mode* ist in *WRIREvent* definiert und kann die Werte *WRIREvent.BASIC*, *WRIREvent.EXTENDED* oder *WRIREvent.FULL* annehmen. Sie stehen für die unter 2.2 beschriebenen Betriebsmodi.

Der Parameter *sensitivitySettings* kann einen von fünf verschiedenen Werten aus der Klasse *IRSensitivitySettings* übergeben werden: *IRSensitivitySettings.WII_LEVEL_1* bis *IRSensitivitySettings.WII_LEVEL_5* entsprechen der niedrigsten bis zur höchsten Empfindlichkeit des IR-Sensors. Genaue Informationen über diese Empfindlichkeitslevel sind nicht bekannt.

4.2 Programm WRTTest

Das Programm wurde geschrieben, um die Werte von *WiiRemoteJ* in der Praxis zu analysieren. Anlass dazu gab die Tatsache, dass bestehende Realisierungen, wie *WiiWhiteboard* 4.3 sich die Bounding Box und verschiedene Empfindlichkeitslevel nicht zunutze machen und nur simple Informationen von der Wiimote beziehungsweise von *WiiRemoteJ* abfragen.

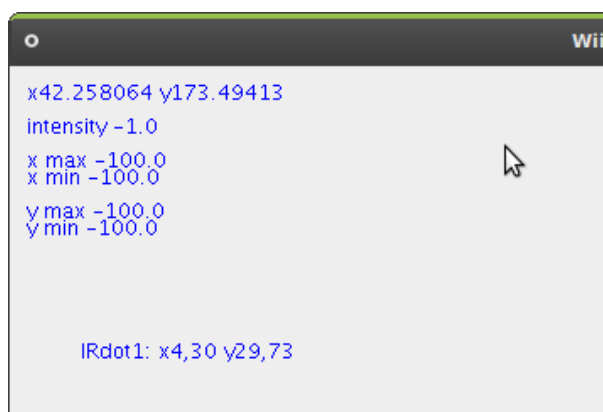


Abbildung 4: Screenshot WRTTest im *EXTENDED* Modus



In [Abbildung 4](#) ist zu sehen, dass die Werte der Bounding Box und der Intensität negative Werte beinhalten. Wie in der JavaDoc von *WiiRemoteJ* beschrieben ist dies gemäß der Implementation korrekt, da diese Werte im *EXTENDED*-Modus nicht übertragen werden.

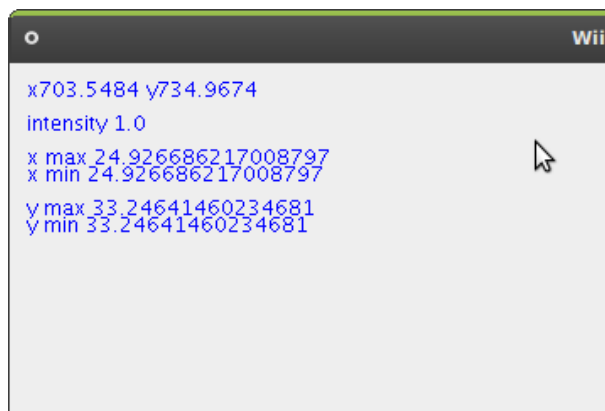


Abbildung 5: Screenshot WRTTest im *FULL* Modus

In [Abbildung 5](#) ist zu sehen, dass sich jedoch auch im *FULL*-Modus keine Besserung zeigt. Es existieren zwar Werte, diese verändern sich jedoch während der Verbindung zur Wiimote nicht.



4.3 Programm WiimoteWhiteboard

Das Programm *WiimoteWhiteboard* stammt von Uwe Schmidt und basiert auf den Quelltexten von Johnny Lee. Aktuell ist derzeit Version 0.9.8. Die Software, welche in Java geschrieben wurde, bietet eine Vielzahl Funktionen, unter anderem zur Maussteuerung mit einer WiiRemote. Dabei werden mehrere WiiRemotes unterstützt, um eine Fläche präziser abzutasten oder tote Winkel zu vermeiden, es werden bis zu vier Positionspunkte verarbeitet und mehrere Monitore oder Beamer unterstützt. Die Projektion der Koordinaten geschieht dabei durch Suns Java Advanced Imaging API [mic09]. Zur Verbindung zu den WiiRemotes wird *BlueCove* und *WiimoteJ* verwendet.

4.3.1 Besonderheit BlueCove

Um BlueCove in einer 64-Bit Umgebung lauffähig zu bekommen, wurde die in *WiiWhiteboard* enthaltene Version durch die aktuellste Version, derzeit 2.1.0, ersetzt. [Blu09] Als problematisch erwies sich dann jedoch die Tatsache, dass die WiiRemote am PCM 1 kommuniziert. PCMs sind vergleichbar mit TCP/IP-Ports. Denn nach JSR-82 sind Benutzerdefinierte Verbindungen, also solche, die keinen reservierten und öffentlich bekannten Port besitzen, erst ab Port 4096 und darüber erlaubt. Die neueste BlueCove-Version hält sich an diese Empfehlung und verbietet die Verbindung über Port 1. Der Versuch wird mit folgender Fehlermeldung bestraft:

```
java.lang.IllegalArgumentException: PCM values restricted by JAR82  
to minimum 4097
```

Um dies zu umgehen kann BlueCove ein Parameter mitgegeben werden, der diese Beschränkung aufhebt.

```
$ java -Dbluecove.jsr82.psm_minimum_off=true -jar WiimoteWhiteboard.jar
```

Genauere Informationen können unter [Janb] abgerufen oder diskutiert werden.

4.3.2 Bedienung WiimoteWhiteboard

Das Programm sucht nach dem Start eigenständig nach verfügbaren WiiRemotes über Bluetooth. Diese müssen, damit sie gefunden werden können in den Discoverable-Mode versetzt werden. Dazu sind die beiden Tasten 1 und 2 an der Oberseite gleichzeitig zu drücken. Die Status-LEDs am unteren Rand beginnen daraufhin schnell zu blinken. Ist eine Verbindung hergestellt, ist nur die linke LED eingeschaltet.

Die WiiRemote ist nun noch unkalibriert und kann noch nicht für Mauseingaben verwendet werden. Um festzustellen, ob überhaupt IR-Punkte aufgenommen werden bietet sich nun ein Klick auf *IR Kamera Monitor* an. Dieser zeigt das Sichtfeld der WiiRemote an und stellt die Punkte als nummerierte Kreise an, damit die Punkte unterscheidbar bleiben, falls ein zweiter IR-Punkte im Sichtfeld auftaucht.

Falls ein IR-Punkt nicht wie in Abbildung 7 zu sehen ist, sollte die IR-Lichtquelle und die WiiRemote geprüft werden. Zudem sollte auf einen Mindestabstand von 15 cm geachtet werden, da bei geringeren Distanzen



Abbildung 6: WiimoteWhiteboard Hauptfenster

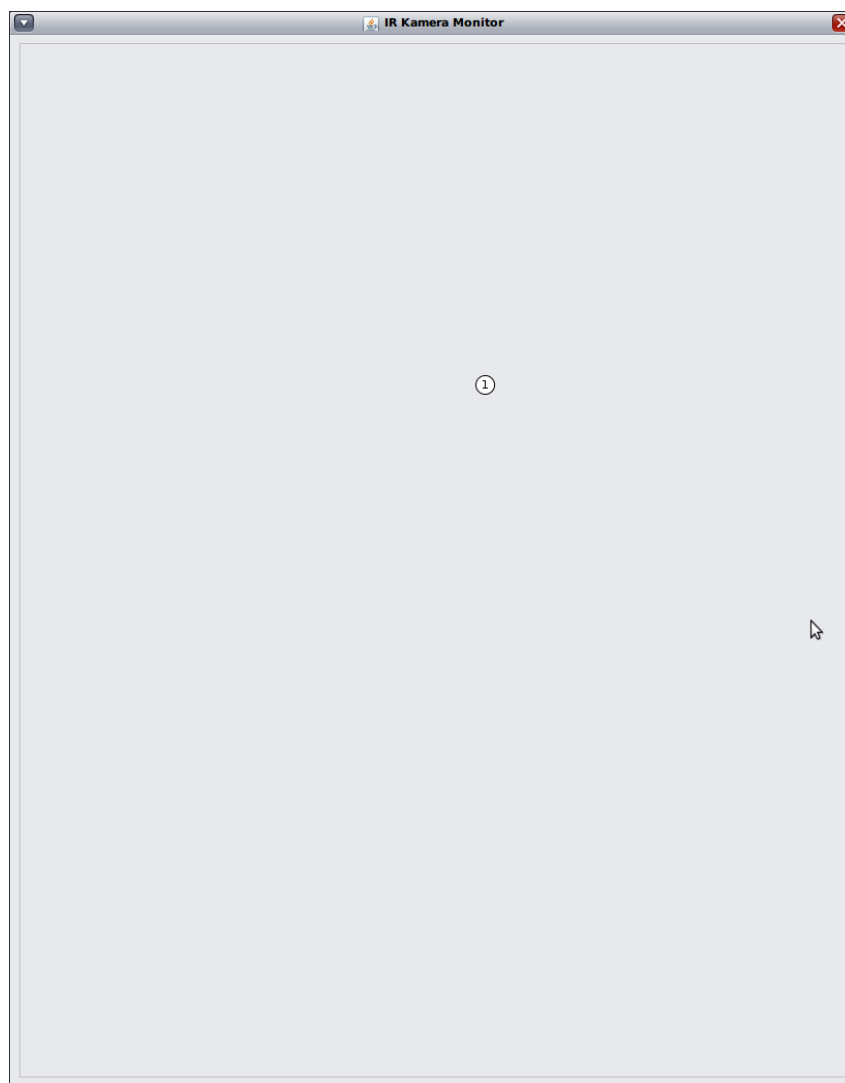


Abbildung 7: WiimoteWhiteboard IR Kamera Monitor

keine klare Position mehr bestimmt werden kann. Dies scheint ein Problem der WiiRemote-Optik zu sein, dies lässt sich jedoch nicht genau bestimmen.

Um die WiiRemote zu kalibrieren ist der Button *Kalibriere* zu betätigen. Er zeigt, wie in Abbildung 8 zu sehen ist, auf dem gewählten Bildschirm einen Kalibrierungsdialog an, der auffordert, die Projektionsfläche an allen vier Ecken mit IR-Punktmarkierungen zu versehen. Mit den vorgestellten IR-Stiften in Kapitel 3.1 ist dazu ein Druck auf den Taster an der entsprechenden Position auf der Wand nötig. Wird ein Punkt erkannt, wird dort, wo vorher ein Kreuz zu sehen ist, ein grüner Haken erscheinen und mit der nächsten Bildschirmecke fortgefahren. Mit den vier erkannten Punkten ist eine Transformation von einem aufgenommenen Punkt auf der Projektionsfläche zu einem Punkt auf dem Bildschirm möglich. Siehe dazu auch Abbildung 9.

Ist die Kalibrierung erfolgreich beendet, stehen alle Funktionen zur Steuerung der Maus zur Verfügung. Nach einer erfolgreichen Kalibrierung wird diese Funktion auch direkt aktiviert; allerdings im *Nur Bewegen*-Modus, um ungewollte Klicks zu vermeiden. In diesem wird die Mausposition nur dem empfangenen IR-Punkt nachgeführt. Aktiviert man den Modus *Bewegen & Klicken* wertet die Software den Empfang eines IR-Punkts als Druck auf die linke Maustaste an der Mausposition der IR-Quelle. Somit sind Klicks und Mausgesten wie

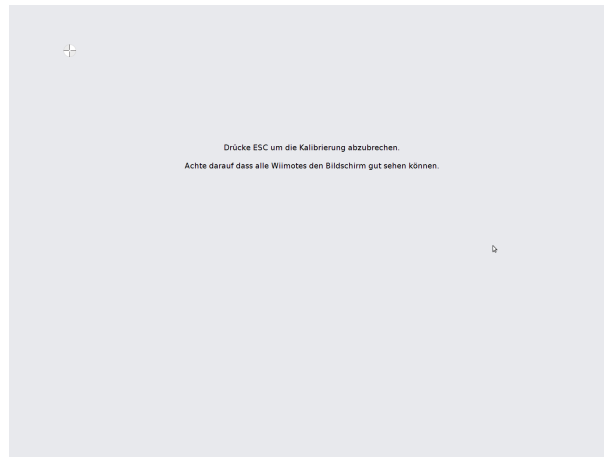


Abbildung 8: WiimoteWhiteboard Kalibrierungsdialog

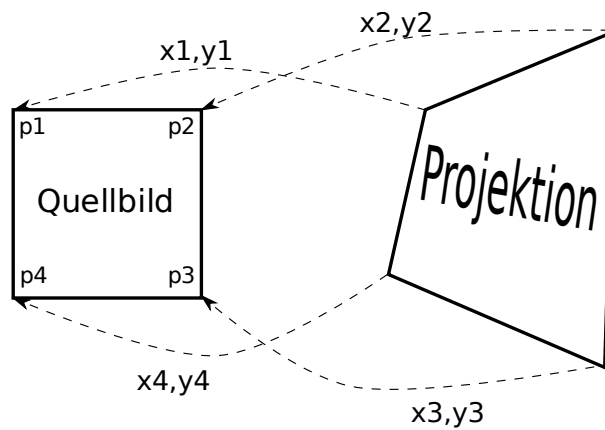


Abbildung 9: Projektionsprinzip

Drag and Drop oder das Markieren von Text problemlos möglich.

Interessant sind an dieser Stelle auch die Kalibrierungsdetails, welche durch einen Klick auf den gleichnamigen Button aufrufbar sind. Sie zeigen an, wie die von der WiiRemote aufgenommene Fläche ausgenutzt wird. In der Abbildung 10 ist zu sehen, dass nur 38% der aufgenommenen Fläche zur Projektion verwendet wird. Je kleiner dieser Anteil ist, desto kleiner ist auch die Präzision der Positionserkennung, da sich die reell projizierte Pixelzahl auf eine kleinere aufgenommene Pixelzahl verteilt. Ein aufgenommener Pixel steht somit für eine größere Zahl projizierte Pixel.

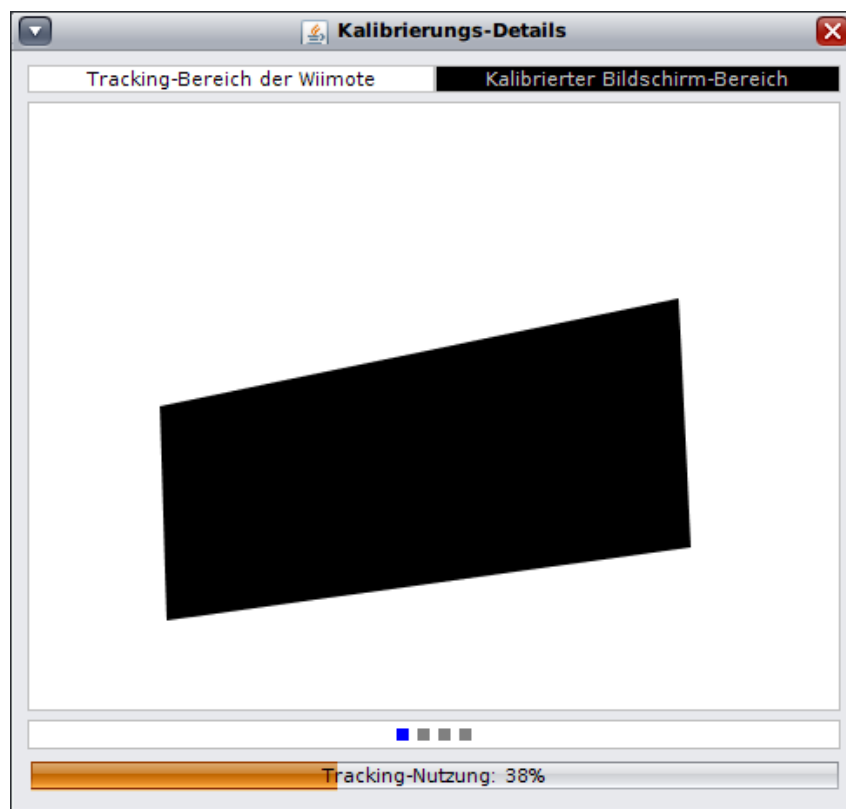


Abbildung 10: WiimoteWhiteboard Kalibrierungsdetails



5 Beobachtungen

5.1 Empfindlichkeit der IR-Kamera

Die Empfindlichkeit der Kamera reicht aus, um die ca. 2,5m mal 2,5m große Projektionsfläche aufzunehmen. Bei kompletter Überschneidung der Projektionsfläche und Sensorfeld der Kamera würde sich eine Genauigkeit von 2,5 Pixel pro Millimeter ergeben.

5.2 Leistung der IR-LEDs

Die Leistung der IR-LEDs reicht nicht aus, um noch etwas wahrzunehmen. Hier besteht noch starkes Verbesserungspotential. Wie in Kapitel 3.1 beschrieben, lassen sich bessere IR-LEDs einsetzen. Es wäre zu testen, ob diese auch die beschriebene Projektionsfläche „versorgen“ können.

5.3 Praxistauglichkeit

Die Versuche haben eine starke Winkelabhängigkeit gezeigt. Bei einer Distanz von bis zu einem halben Meter fällt dies nicht sonderlich auf, da die Reflektion auf der getesteten Fläche ausreicht.

Wand Auf der getesteten Wand befand sich weiß gestrichene Rauhfaserapete. Die Wiimote befand sich auf einem Stativ in einem Abstand von ca. 1m. Hierbei waren schon Winkelabhängigkeiten bemerkbar.

Leinwand Die Beamerleinwand versprach gute Resultate da sie leicht reflektiert. Leider bestätigte sich dies nicht, da die Oberfläche mehr darauf ausgelegt ist, das einfallende Licht zu streuen. Damit wurde der IR-Lichtkegel so weit vertret, dass nicht mehr genau bei der Wiimote ankam. Getestet wurde bei einer Distanz von ca. 4m. Dreht man den IR-Stift um und richtet die LED direkt auf die Wiimote reicht das Signal. In dieser umständlichen Haltung funktioniert es recht gut und überraschend präzise.

DIN A4 Blatt Das Blatt befand sich auf einem Tisch, die Wiimote war von oben auf einem Stativ auf dieses Blatt gerichtet. Bei dieser Konstellation gab es keine Probleme.

Bildschirm (verspiegelt) Der Verspiegelte Laptop-Bildschirm stellte kein Problem für die Wiimote dar. Es war sogar der Effekt zu beobachten, dass, wenn der IR-Stift sich ca. 3cm vor dem Bildschirm befand, von der Wiimote zwei Punkte wahrgenommen wurden. Der größere von beiden war der gespiegelte Punkt, ein kleinerer „Satellit“ stellte sich als Licht aus der seitlichen Keule der IR-LED heraus. Beim Verringern der Distanz fasste die Wiimote beide Punkte zu einem zusammen. Die Punkte müssen also einen Mindestabstand zueinander haben.

Dies passt zum Aufbau der Sensorbar von Nintendo, in dem in geringem Abstand an jeder Seite fünf IR-LEDs angebracht sind. Würden diese als Einzelpunkte wahrgenommen werden, wäre ein zuverlässiger Nahbertrieb nicht möglich.

Wird der Abstand vergrößert, wird die Intensität zu klein und es wird kein Punkt mehr wahrgenommen. Die seitliche Abstrahlung würde jetzt zwar genügen, jedoch befindet sich der Stift nun im falschen Winkel.

Bildschirm Matt Auf einem matten Bildschirm ergeben sich in etwa die gleichen Nachteile wie auf der Projektionsfläche des Beamers. Ein Test mit Cellophan-Folie brachte keine nennenswerten Erfolge.



6 Fazit

6.1 WiiRemote als SmartBoard

Bei einer Festinstallation oder einem stark eingespielten Aufbau, bei der die WiiRemote immer genau die gleiche Position einnimmt sollte es möglich sein, geeignete IR-Lichtquellen vorausgesetzt, die WiiRemote als SmartBoard einzusetzen. Dann ist die WiiRemote eine kostengünstige Alternative zu den sonst sehr teuren Alternativen, welche jedoch ausgereifter sind und die erwähnten Nachteile oft nicht haben. Es sollte je nach Einsatzgebiet entschieden werden, was gebraucht wird.

Wechselt die Position häufiger, so ist zumindest bei einer Präsentation erheblich mehr Zeit und ein Plan B einzukalkulieren, da herumexperimentiert werden muss, bis die WiiRemote alle Ecken der Leinwand aufnimmt. Ein erheblicher Vorteil wären hierbei vier kleine IR-LEDs, die mit einer Knopfzelle betrieben an den Ecken einer Leinwand angebracht würden. Zum Beispiel könnte man vier Wäscheklammern, an die man zusätzlich einen Magnet klebt herhalten. So könnte man sie an freistehende Leinwände klemmen und hätte an Whiteboards Magnethaftung.

Bei einer Präsentation musste ich mich ersteinmal daran gewöhnen, solche für einen Mausbenutzer starke Bewegungen zur Bedienung eines Computers zu vollführen. Steht man so nah an einer Leinwand, neigt man eher dazu, sich von ihr ein paar Schritte zu entfernen, um den Überblick nicht zu verlieren. Zur Eingabe muss man jedoch wieder an die Leinwand herantreten. Dies endete damit, dass ich mich wieder wie gewohnt an die Maus gesetzt habe, um etwas präsentieren.

Ein anderer Ansatz wäre es, die WiiRemote in der Hand zu halten und die Position über ein paar festinstallierte IR-LEDs zu bestimmen. Die Nintendo SensorBar, welche zehn davon an zwei Enden einer Stange besitzt, dient nur der zweidimensionalen Positionsbestimmung. Bei geschickter Anordnung ließe sich auch eine dreidimensionale Position ermitteln. Dies könnte hilfreich sein, um dem Bediener den Freiraum für einen persönlich angenehmen Abstand zu lassen und zusätzlich dreidimensionale Gesten einzubauen. Bei gedrückter Taste und Bewegung zur Leinwand herauszoomen (reindrücken), bei einer Bewegung von der Leinwand weg heranzoomen (ranholen).

6.2 Ist WiimoteWhiteboard praxistauglich?

Die Software ist meines Erachtens ein vielversprechender Anfang, der alles mitliefert, was man zum Start braucht. Feineinstellungen lassen sich jedoch nur im Quelltext ausführen. So könnte man beispielsweise den Schwellpunkt, bei dem zwei aufgenommene IR-Punkte als einer wahrgenommen werden, einstellbar sein. Des Weiteren bietet es viel Flexibilität durch die bereits erwähnte Multiscreen und Multi-WiiRemote-Unterstützung.

Als problematisch stellte sich die Multitouch-Funktion heraus. Die WiiRemote kann zwar bis zu vier IR-Punkte aufnehmen, jedoch mangelt es an standardisierten Möglichkeiten, diese Punkte einem Programm mitzuteilen. Es gibt ein Netzwerkprotokoll, das auf XML Basis positionsdaten versendet, jedoch sind das alles Workarounds. Windows 7 soll eine native API dafür erst liefern.[Gol08] Wahrscheinlich wird dann das Mouse-Event nicht mehr als einzelne xy-Koordinate sondern als Array aus Koordinaten behandelt werden können.

6.3 Die Zukunft der Computereingabe

Es ist abzusehen, dass Computereingaben noch in wesentlich mehr Bereiche vordringen werden, als es jetzt bereits der Fall ist. Darunter werden auch lebenskritische Anwendungen in der Medizin oder in der Fahrzeugtechnik sein - vom Fahrstuhl bis zur Raumkapsel. Bisher ist dies jedoch immer mit abstraktem



Denken verbunden. Es muss überlegt werden „Wie bringe ich das was ich will dem Computer bei?“. Ziel wird es sein, diese Spalte zu schließen. Gestensteuerung, Sprachbefehle oder allein die Möglichkeit, alle Gliedmaßen, die dem Menschen zur Verfügung stehen, zu nutzen, um eine Aufgabe mit dem Computer zu erledigen werden dazu beitragen.



Webseiten

- [Blu09] BlueCove: *BlueCove JSR 82 project*. <http://www.bluecove.org/>. Version: 2009. – [Online; Stand 27. Januar 2009]
- [EDEK09] ELKO Das Elektronik KOMPendium das: *Bluetooth 1.0 / 1.1 / 1.2*. <http://www.elektronik-kompodium.de/sites/kom/0803301.htm>. Version: 2009. – [Online; Stand 24. Februar 2009]
- [Gol08] Golem.de: *Windows 7 mit Multitouch*. <http://www.golem.de/0805/59999.html>. Version: 2008. – [Online; Stand 28. Mai 2008]
- [Jana] Janzen, Sebastian (Hrsg.): *Meine Infrarotstifte*. <http://wp.janzen.it/2009/01/18/meine-infrarotstifte>, Abruf: 27. Januar 2009
- [Janb] Janzen, Sebastian (Hrsg.): *WiiRemoteJ 1.5 mit Bluecove >= 2.1.0*. <http://wp.janzen.it/2009/01/18/wiiremotej-15-mit-bluecove-203>, Abruf: 24. Februar 2009
- [Jü09] Jürss, Udo: *ATM18 Wii Projekt*. <http://www.cczwei-forum.de/cc2/thread.php?postid=18607>. Version: 2009. – [Online; Stand 27. Januar 2009]
- [mic09] microsystems, Sun: *Java Advanced Imaging (JAI) API*. <http://java.sun.com/javase/technologies/desktop/media/jai/>. Version: 2009. – [Online; Stand 24. Februar 2009]
- [Tea09] Team, Wii Homebrew P.: *Homebrew Starter Paket*. <http://www.wii-homebrew.com/homebrew-einsteiger/>. Version: 2009. – [Online; Stand 15. Februar 2009]
- [Wii09a] WiiLi.org: *Johnny Chung Lee Projekts for Wii*. <http://www.cs.cmu.edu/~johnny/projects/wii/>. Version: 2009. – [Online; Stand 25. Januar 2009]
- [Wii09b] WiiLi.org: *WiiLi.org a GNU/Linux port for the Nintendo Wii*. <http://www.wiili.org/index.php?title=Wiiote&oldid=48229>. Version: 2009. – [Online; Stand 25. Januar 2009]
- [Wii09c] WiiRemoteJ: *WiiRemoteJ Library*. <http://www.world-of-cha0s.hostrocket.com/WiiRemoteJ/>. Version: 2009. – [Online; Stand 27. Januar 2009]
- [Wik09a] Wiki, Wiimoteproject: *IR Sensor*. http://wiki.wiimoteproject.com/IR_Sensor. Version: 2009. – [Online; Stand 27. Januar 2009]
- [Wik09b] Wikipedia: *Wii-Fernbedienung* — *Wikipedia, Die freie Enzyklopädie*. <http://de.wikipedia.org/w/index.php?title=Wii-Fernbedienung&oldid=55490183>. Version: 2009. – [Online; Stand 24. Januar 2009]

Bücher

- [Dev09] Devices, Analog: *ADXL330: Small Low Power 3-Axis ±3g iMEMS[®] Accelerometer*, 2009. <http://www.analog.com/en/mems-and-sensors/imems-accelerometers/adxl330/products/product.html>. – [Online; Stand 27. Januar 2009] 5
- [Kin09] Kingbright: *IR-LED 5MM Typ L-53F3C*, 2009. http://www.produktinfo.conrad.com/datenblaetter/150000-174999/154447-da-01-en-IR-LED_5MM_TYP_L-53F3C.pdf. – Vertrieben durch Conrad Elektronik <http://www.conrad.de>
- [Vis09] Vishay: *High Power Infrared Emitting Diode*, 2009. <http://www.vishay.com/docs/81011/tsal6400.pdf> 7



Listing 1: Programm WRTest

```
1 import java.awt.*;
2 import java.text.DecimalFormat;

4 import javax.swing.JFrame;
5 import javax.swing.JPanel;

7 import wiiremotej.*;

9 /**
10  * Programm to check Data from <i>WiiRemoteJ</i> library.
11  * To get more output set DEBUG field to <i>>true</i>.
12  * @author Sebastian Janzen <sebastian@janzen.it>
13  *
14  */
15 public class WRTest extends WiiRemoteAdapter {

17     private WiiRemote remote;
18     private static double irIntensity;
19     protected static boolean gotIRDots;
20     private static double irX, irY;
21     private static double irMaxX, irMinX;
22     private static double irMaxY, irMinY;
23     private static JFrame graphFrame;
24     private static JPanel graph;

26     private static final boolean DEBUG = false;

28     /**
29     * Main Program to init JFrame and search/init Wiimote
30     *
31     * @param args
32     *         from command line
33     */
34     public static void main(String args[]) {
35         if (DEBUG) {
36             WiiRemoteJ.setConsoleLoggingAll();
37         }

39         irX = irY = 0;
40         irMaxX = irMaxY = 0;
41         irMinX = irMinY = 0;

43         try {
44             graphFrame = new JFrame();
45             graphFrame.setTitle("Wii Remote IR Data");
46             graphFrame.setSize(800, 600);
47             graphFrame.setResizable(false);

49             graph = new JPanel() {
50                 private static final long serialVersionUID = -6250598591784476377L;

52                 public void paintComponent(Graphics graphics) {
53                     // Clean up canvas
54                     graphics.clearRect(0, 0, 800, 600);
55                     graphics.setColor(Color.BLUE);

57                     // Calculate positions
58                     float x, y;
```



```
59     double myIrX = irX;
60     double myIrY = irY;
61     int width, height;
62     x = 10 + (float) (myIrX * 7.5); // ca. *8 (Fensterbreite)
63     y = 10 + (float) (myIrY * 5.5); // ca. *6 (Fensterhoehe)

65     // Bounding Box
66     width = (int) (irMaxX - irMinX);
67     height = (int) (irMaxY - irMinY);

69     // Graphics2D bec of double-pos printing
70     Graphics2D g2 = (Graphics2D) graphics;
71     DecimalFormat df = new DecimalFormat("0.00");

73     g2.drawString("IRdot1: x" + df.format(irX) + " y"
74         + df.format(irY), x, y);

76     g2.drawRect((int) (irMinX) + 1, (int) (irMinY) + 1, width,
77         height);

79     g2.drawString("x" + x + " y" + y, 10, 20);
80     g2.drawString("intensity " + irIntensity, 10, 40);
81     g2.drawString("x max " + irMaxX, 10, 60);
82     g2.drawString("x min " + irMinX, 10, 70);
83     g2.drawString("y max " + irMaxY, 10, 90);
84     g2.drawString("y min " + irMinY, 10, 100);

86     }
87     };
88     graphFrame.add(graph);
89     graphFrame.setVisible(true);

91     // Find and connect to a Wii Remote
92     WiiRemote remote = WiiRemoteJ.findRemote();
93     remote.addWiiRemoteListener(new WRTest(remote));
94     remote.setAccelerometerEnabled(false);
95     remote.setIRSensorEnabled(true, WRIEvent.EXTENDED,
96         IRSensitivitySettings.WII_LEVEL_4);
97     remote.setUseMouse(true);

99     } catch (Exception e) {
100         e.printStackTrace();
101     }
102 }

104 public WRTest(WiiRemote remote) {
105     this.remote = remote;
106 }

108 public void disconnected() {
109     System.out.println("Remote disconnected... Please Wii again.");
110     System.exit(0);
111 }

113 public void statusReported(WRStatusEvent evt) {
114     System.out.println("Battery level: " + (double) evt.getBatteryLevel()
115         / 2 + "%");
116     System.out.println("Continuous: " + evt.isContinuousEnabled());
117     System.out
118         .println("Remote continuous: " + remote.isContinuousEnabled());
```



```
119 }
121 public void IRInputReceived(WRIREvent evt) {
122     for (IRLight light : evt.getIRLights()) {
123         if (light != null) {
124             if (DEBUG) {
125                 System.out.println("X: " + light.getX());
126                 System.out.println("Y: " + light.getY());
127             }
129             irX = light.getX() * 100;
130             irY = light.getY() * 100;
131             irMaxX = light.getXMax() * 100;
132             irMinX = light.getXMin() * 100;
133             irMaxY = light.getYMax() * 100;
134             irMinY = light.getYMin() * 100;
135             irIntensity = light.getIntensity();
136             gotIRDots = true;
137             graph.repaint();
139         }
140     }
142 }
144 public void accelerationInputReceived(WRAccelerationEvent evt) {
145     System.out.println("Not implemented");
146 }
148 public void extensionInputReceived(WRExtensionEvent evt) {
149     System.out.println("Not implemented");
150 }
152 public void extensionConnected(WiiRemoteExtension extension) {
153     System.out.println("Not implemented");
154 }
156 public void extensionPartiallyInserted() {
157     System.out.println("Not implemented - and interesting!");
158 }
160 public void extensionUnknown() {
161     System.out
162         .println("Extension unknown. Did you try to plug in a toaster or something?");
163 }
165 public void extensionDisconnected(WiiRemoteExtension extension) {
166     System.out.println("Extension has gone.");
167 }
169 public void buttonInputReceived(WRButtonEvent evt) {
170     System.out.println("Great, you can push buttons! Try another one ...");
171 }
173 }
```