Air Wiitar, Teilaspekt Steuerung

Student: Yannick Rietz - Matrikelnummer: 2084647 - AVPRG Projekt Dokumentation

Einleitung

Ziel unseres Projektes war es, es dem Benutzer unseres Qt-Programms zu ermöglichen, mit Hilfe der Wii-Fernbedienung einen (Gitarren-)Synthesizer zu steuern, nur indem er Bewegungen macht wie beim Spielen einer Gitarre. Auf den folgenden Seiten werde ich näher auf meinen Teilaspekt des Projektes eingehen, nämlich auf die Steuerung. Ich habe zunächst nach einer Library gesucht, die für mich die Daten der Sensoren des Controllers ausliest. Anschließend musste ich mich eingehend mit diesen Daten beschäftigen und herauszufinden, was sich zuverlässig aus diesen folgern lässt. Da sich die Daten als sehr ungenau herausgestellt haben, war ich relativ eingeschränkt bei der Erkennung von Gesten und habe für eine bessere Steuerung zwei Controller-Tasten eingebunden.

Empfangen und Auslesen der Controller-Daten

Die Wii-Fernbedienung arbeitet mit Bluetooth und kann wie andere Bluetooth Geräte einfach mit dem PC gekoppelt werden. Das Erkennen der Wii-Fernbedienung und das Auslesen der Werte der Sensoren wollte ich einer bereits bestehenden Library überlassen. Ich konnte mehrere entsprechende C-Librarys finden, allerdings war deren offizielle Dokumentation häufig aufgrund ihres Alters nicht mehr aufzufinden. Die Wii-Fernbedienung wird zwar noch hergestellt, ist allerdings auch schon seit zehn Jahren auf dem Markt. Dementsprechend ist die Hochzeit der Experimente mit diesem Controller längst vorrüber. Einige dieser Librarys ließen sich nicht (mehr) kompilieren, andere konnten keine Verbindung mit meinen Wii-Fernbedienungen herstellen. Letztendlich funktioniert hat für mich eine Library mit dem klingenden Namen „WiiYourself!“. Deren offizielle Internetseite ist allerdings schon länger nicht mehr erreichbar und ich musste für den Download auf das Internetarchiv Archive.org zurückgreifen. WiiYourself arbeitet mit Polling, das heißt es wird in kurzen Abständen überprüft, ob bereits neue Daten zur Verfügung stehen. In diesem Fall wird dann ein Event ausgelöst und die Daten verwendet.

Ich habe für das Polling eine Klasse `WiiThread` implementiert, die von QThread erbt, und über eine Membervariable vom Typ `wiimote` aus der Library verfügt. Diese Klasse enthält nur eine Methode `run`. Innerhalb dieser wird zunächst solange versucht, eine Verbindung zu einer Wii-Fernbedienung herzustellen, bis dies erfolgreich war. Es folgt eine Endlosschleife, in der das Polling durchgeführt wird. Die für uns relevanten Daten werden dort in einem Struct verpackt und über das SIGNAL/SLOT System von Qt an meine Klasse WiiAnalyser zur Weiterverarbeitung gesendet.

Abb. 1: Wii-Fernbedienung mit Nunchuk

Probleme mit der Genauigkeit der Daten

Die Wii-Fernbedienung verfügt über drei Sensor-Typen: Ein dreiachsiges Gyroskop zur Erkennung von Rotation, ein dreiachsiges Accelerometer (Beschleunigungssensor) zum Erkennen von Beschleunigung und eine Infrarotkamera, die zum Erkennen der Position von zwei Infrarot-LEDs über dem Fernsehers genutzt wird. Über ein Kabel lässt sich ein kleinerer Controller anschließen, der sogenannte Nunchuk (Abb.1). Dieser Verfügt über ein weiteres Paar Sensoren für die zweite Hand: Gyroskop und Beschleunigungssensor. Wir möchten zwei Gesten erkennen: Eine Auf- und Ab-Bewegung soll das Anschlagen der Saiten repräsentieren. Eine Bewegung auf der waagerechten Achse soll das Ändern der Tonhöhe / des Akkords repräsentieren. Für uns interessant sind also nur die Daten der Beschleunigungssensoren, da Rotation oder die relative Position zu einem Objekt im Raum für uns nicht interessant sind.

Unter perfekten Bedingungen und mit perfekter Hardware könnte man Positionsänderungen der Wii-Fernbedienung nur anhand der Daten der Beschleunigungssensors millimetergenau bestimmen. Tatsächlich sind Beschleunigungssensoren jedoch noch zu anfällig gegenüber Rauschen und Verzerrungen der Daten, damit dies wirklich gut funktioniert. Besonders die zehn Jahre alte Hardware der Wii Fernbedienung erfüllt diese Ansprüche natürlich nicht. Man darf dabei auch nicht vergessen, dass es sich hierbei im Grunde genommen um Spielzeug handelt.

Um aus der aktuellen Beschleunigung die aktuelle Geschwindigkeit des Sensors zu errechnen kann man die Daten integrieren. Um aus der Geschwindigkeit zu berechnen, wie groß die Positionsänderung ist, kann man das Ergebnis noch einmal integrieren (Abb.2). Bei diesem Vorgehen werden sämtliche Ungenauigkeiten nicht nur vervielfacht, sondern auch mit der Zeit addiert. Dies kann dazu führen, dass schon nach ein paar Sekunden nicht mehr davon ausgegangen werden kann, dass die Position genau getrackt wurde. Oder im Falle der Wii-Fernbedienung noch schneller. Insbesondere kann nicht zwischen Entschleunigung und Beschleunigung in die Gegenrichtung unterschieden werden. Für unser Projekt bedeutet dies, dass zum Beispiel die Auswahl der zu spielenden Töne nicht allein durch Bewegen des Controllers entlang des gedachten Gitarrenhalses möglich ist, da sich derartige Positionsänderungen auf Dauer nicht verfolgen lassen nur anhand der Beschleunigungsdaten.

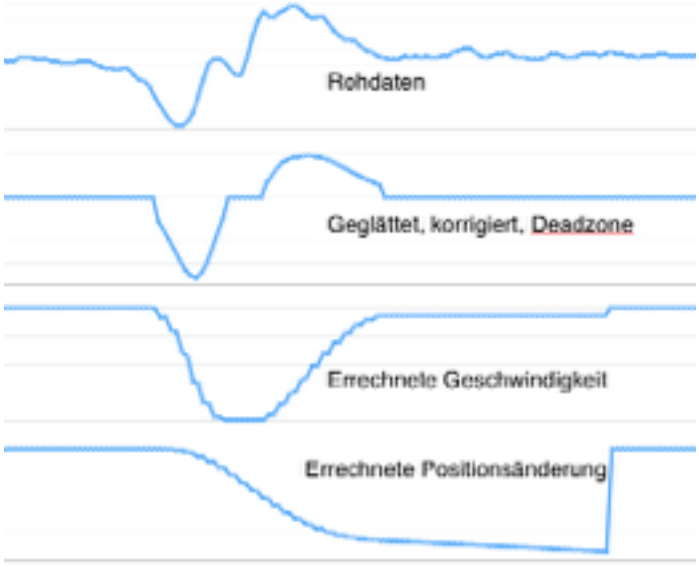

Abb. 2: Visualisierung der empfangenen Daten

Meine Umsetzung

Ich habe mich entschieden die Tasten der Controller mit in die Gesten einzubinden, um etwas unabhängiger von der Ungenauigkeit des Beschleunigungssensors zu werden. Außerdem habe ich einige simple Algorithmen implementiert, um die Fehler in den Daten etwas auszugleichen. Dies geschieht in meiner Klasse `WiiAnalyser`, die erkannte Gesten über das SEND/SLOT System von Qt an den darüber liegenden Layer sendet. Der Einfachheit halber betrachte ich für die Griffhand nur die Horizontale Achse und für die Anschlangshand nur die Vertikale Achse der Beschleunigungssensoren.

Die WiiYourself! Library stellt eine Variable `Acceleration.Orientation.UpdateAge` zur Verfügung. Mithilfe dieser lässt sich sagen, ob der Controller gerade unbewegt ist, das heißt, die Werte alle Achsen des Accelerometers addiert ergeben 1, was der Schwerkraft der Erde entsprechen soll. Ist der Controller also gerade unbewegt, kann man anhand der Richtung dieses Schwerkraftvektors die Ausrichtung des Controllers feststellen und speichern, sodass man später die Einwirkung der Schwerkraft aus den Daten herausrechnen kann. Ich filtere die Daten mit einem Tiefpass, um sie leicht zu glätten und bestimme eine Deadzone, innerhalb welcher ich den Ausschlag als 0 annehme, um Rauschen zu unterdrücken. Außerdem habe ich eine Membervariable `BIAS` erstellt für jeden Beschleunigungssensor, sollte er eine starke Tendenz in eine bestimmte Richtung zeigen.

Für die Griffhand, die zur Auswahl der Tonhöhe bewegt werden soll, habe ich entschieden, dass der Knopf ständig gedrückt gehalten werden muss, während man „spielt“, wie bei einer richtigen Gitarre, bei der man die Saiten auf die Bünde drückt. Möchte man nun den Akkord wechseln, so lässt man die Taste los. Die erste größere Positionsänderung, die jetzt registriert wird, interpretiert mein Programm als Geste. Man kann also je Geste nur einen Schritt nach oben oder unten gehen, nicht mehr. Solange der Knopf gedrückt gehalten wird, betrachte ich die Position als 0. Erst, wenn der Knopf losgelassen wird, wrden Positionsänderungen berechnet. Auf diese Weise versuche ich zu umgehen, dass die Ergebnisse der doppelten Integration nach so kurzer Zeit unbrauchbar werden.

Für die Anschlagshand verzichte ich auf die doppelte Integration und arbeite mit noch konkreteren Annahmen: Ich gehe davon aus, dass, wenn der Knopf des Controllers gedrück ist, jede Bewegung auch einen Anschlag der virtuellen Saiten auslösen soll. Das heißt, ein Ausschlag der Beschleunigung ins Positive mit anschließendem Ausschlag ins Negative wird bei gedrücktem Knopf als Aufwärts-Anschlag interpretiert. Wird der Knopf nun nicht losgelassen, wird der nächste Ausschlag ins Positive als Anschlag in die Gegenrichtung interpretiert und so weiter. Dabei werden jeweils die Extrema des Ausschlags ermittelt und an den darüber liegenden Layer gesendet, um dort nicht nur die Richtung, sondern auch die Stärke des Anschlags nutzen zu können.