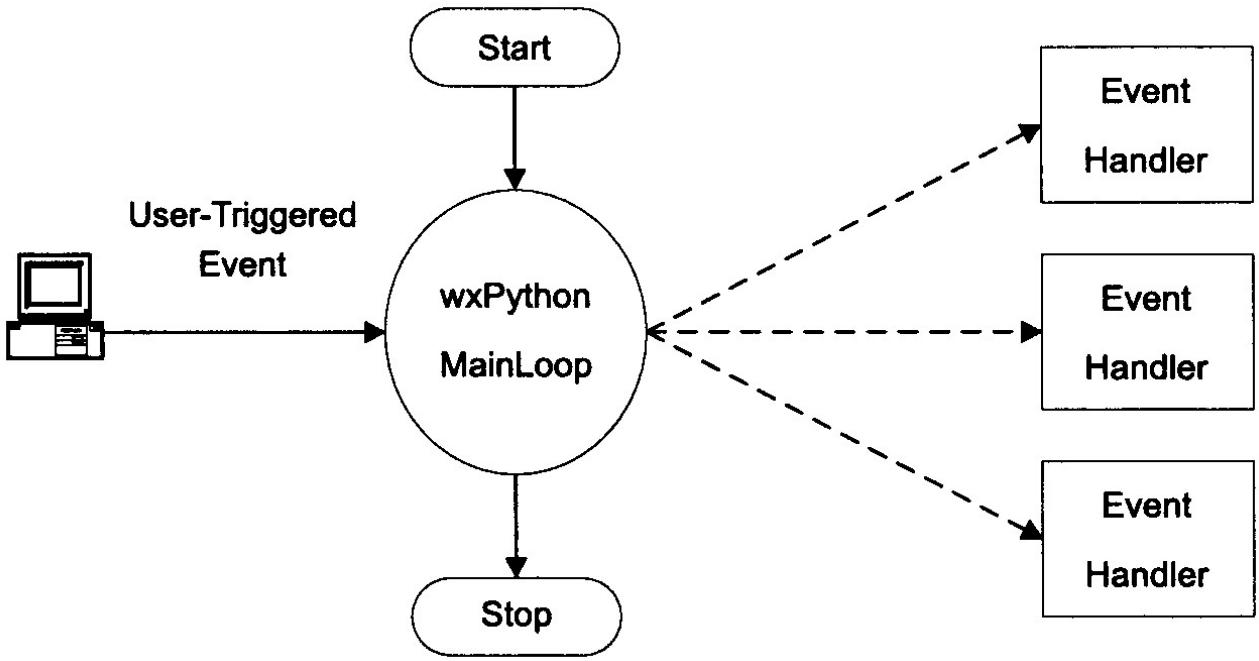
Ereignis-gesteuerte Programmierung

„***Event-driven Programming***" ist ein grundsätzlich anderes Konzept als das traditionelle „EVA-Prinzip" [Eingabe-Verarbeitung-Ausgabe], bei der ein Anwender einen Bearbeitungsprozess anstößt und dann darauf wartet, dass dieser Bearbeitungsprozess fertig ist und er eine Ausgabe erhält.
Bei der Ereignis-gesteuerten Programmierung reagiert ein Programm auf Ereignisse, die für es relevant sind und reagiert dann mit einer Bearbeitung, die schließlich zu einer Ausgabe führen kann. Tritt kein Ereignis auf, wartet das Programm. Das geschieht für den Benutzer scheinbar ohne zu arbeiten.

Kein vorgegebener Programmablauf

Das Ereignis-gesteuertes Programm beschreibt keinen vorgegebenen Ablauf von Arbeitsschritten, da es in der Regel auf viele mögliche Ereignisse reagieren kann. Dadurch bekommt der Benutzer die Möglichkeit, die Steuerung der Arbeit zu bestimmen und in vielen Fällen auch während eines Bearbeitungsschrittes einzugreifen.
Im Buch von Rappin & Dunn[1] wird das Laufzeitverhalten einer in wxPython laufenden Anwendung schematisch so dargestellt:



Einmal gestartet, wartet das Programm auf Ereignisse, zu denen es die entsprechenden Ereignisbehandlungen anbietet. Selbstverständlich sollte mindestens einer der eingezeichneten „***Event-Handler***" die Möglichkeit zum ordnungsgemäßen Beenden des Programms bieten.

Die Inaktivität ist relativ

Während das Programm scheinbar ohne Aktivität ist und möglicherweise auch tatsächlich gerade keine Ereignisse verarbeiten muss, laufen dennoch im Hintergrund Prozesse ab, die man bildlich mit *„Augen und Ohren offen halten"* bezeichnen könnte: Das Programm

1 Noel Rappin, Robin Dunn: wxPython in Action; Manning ;ISBN 1-932394-62-1.
Der hier vorliegende Text orientiert sich am entsprechenden Abschnitt in deren Buch, das Diagramm entstammt dem Buch.

muss periodisch prüfen, ob eines der relevanten Ereignisse auftritt (aufgetreten ist).
Für diesen Vorgang gibt es zwei Varianten, von denen die erste durch wxPython[1] eingesetzt wird:

- Alle auftretenden Ereignisse werden in eine programmweite Warteschlange gestellt und von dort aus nacheinander deren Verarbeitung angestoßen.
- Alternativ fragt das Programm mögliche Quellen von Ereignissen periodisch ab, ob sie etwas zu melden haben.

Aufgaben für den Programmentwickler

Orientieren wir uns an der wxPython-Variante, dann wird deutlich, dass nicht der Programmentwickler die Ereignisse in die Warteschlange stellt, sondern der (oder die) Anwender. Daher muss sich der Programmentwickler Gedanken darüber machen, welche Beziehungen unterschiedliche Ereignisse und Arbeitsschritte haben können.
Ein elementares Beispiel ist die Anforderung, den Benutzer darauf aufmerksam zu machen, dass Bearbeitungsergebnisse möglicherweise gespeichert werden sollten, bevor das Programm geschlossen wird. Es sollte auch eine Möglichkeit geben, dass der Benutzer erfährt und darauf reagieren kann, dass eine Teilbearbeitung noch läuft usw.
Je komplizierter Software wird, desto weniger ist es allein das Problem des „*dümmsten anzunehmenden Users*", um das es geht, sondern die angemessene Benutzbarkeit („*Usability*") der Software.

Hohe Kohäsion und lose Kopplung

Hohe Kohäsion und lose Kopplung gehören zu den grundlegenden Anforderungen an einen guten objektorientierten Software-Entwurf und werden durch die Trennung zwischen Ereignisquelle und Ereignisverarbeitung unterstützt.
Das hat aber auch zur Folge, dass es sein kann, dass es nicht ganz einfach ist, den Kontrollfluss in einem Programm zu verfolgen. Wenn ein Button-Klick an einen beliebigen Ort gesandt werden kann, um dort verarbeitet zu werden, führt das zwar durch die lose Kopplung zu einer leichten Modifizierbarkeit von Software, macht aber das Verfolgen von Aktionen[2] schwieriger.
Das wichtigste grundlegende Konzept der objektorientierten Softwareentwicklung für grafische Benutzeroberflächen, das „***Model View Controller***" - Konzept ist ein gutes Beispiel dafür. Im Kurs[3] sollte es daher umgesetzt und hinterfragt werden.

1 Stichworte für Java: "***Delegation Event Model***" bzw. "***Delegation Based Event Handling***".
Empfänger sind die (Event-) ***Listener***. Listener sind Objekte, die vom Programm erzeugt werden und bei der Event-Quelle angemeldet sein müssen. Nur an diese registrierten Listener-Objekte wird beim Auftreten des Ereignisses eine Botschaft (***message***) geschickt.

2 Besonders bei der Berücksichtigung nebenläufiger Prozesse (***threads***) ist viel zusätzlicher Planungsaufwand durch den Programmentwickler notwendig, um schwerwiegende Fehlersituationen zu vermeiden.

3 Bei einem Kurs mit wxPython hilft das Demo-Tool beim Einstieg in die GUI-Entwicklung.
Bei einem Java-Kurs sollte man beim Einstieg in die GUI-Entwicklung auf das Werkzeug javaeditor wechseln. Hier werden automatisch die grundlegenden Objekte und Methoden eingebaut und man kann gut zeigen, wie man mit MVC die Aufgaben trennen kann und weswegen man bei einfacher Software gern auf einen gesonderten Controller verzichtet.