App-Entwicklung mit Python, Kivy und Plyer

Wolfschmitt, André   
Fakultät Angewandte Naturwissenschaften  
Hochschule für angewandte Wissenschaften CoburgCoburg, Germany  
[woan1500@stud.hs-coburg.de](mailto:woan1500@stud.hs-coburg.de)

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract*. (*Abstract*)

Keywords—component, formatting, style, styling, insert (key words)

# Einleitung

Im Rahmen der zu Grunde liegenden Bachelorarbeit (vgl. [10]), wird ein Projekt des Studiengangs „Technische Physik“ der Hochschule Coburg aus dem Sommersemester 2018 aufgegriffen. Ziel dieses Projekts bildet die Konstruktion einer sogenannten „Science Escape Box“. Um an das Innere der Box zu gelangen, müssen Teilnehmer naturwissenschaftliche Aufgaben und korrespondierende Codes lösen. Genauere Informationen zu Hintergrund und Verlauf der Projektarbeit sind in [10] sowie im hochschulinternen Projektbericht festgehalten.

Motivation und Zielsetzung der Bachelorarbeit bildet die Erweiterung des bereits bestehenden Spielgeschehens um eine digitale Komponente. Zu diesem Zweck, wird ein Tablet in das Projekt integriert, welches sich ergänzend in den Ablauf einfügt und das bisherige Spiel somit nicht ersetzt (vgl. [10]).

Da die Abschlussarbeit in einem naturwissenschaftlichen Studiengang entstanden ist, entzieht sich die Kenntnis und Einhaltung gängiger Architektur- und Modellierungsstrukturen von Software, wie z.B. dem Model-View-Controller-Prinzip, dem Anforderungsniveau. Dennoch ist im Zuge der Arbeit die Entscheidung für eine bestimmte Programmierumgebung, sowie den grundlegenden Aufbau der Software zu treffen. So wird die Programmiersprache Python zur Implementierung der Logik eingesetzt und für die Gestaltung der Benutzeroberfläche auf das Framework „Kivy“ zurückgegriffen. Schließlich verschafft die Nutzung der Bibliothek „Plyer“ Zugriff auf Hardware-interne Features, wie z.B. Sensoren des Tablets.

Das vorliegende Paper beschäftigt sich mit den Bewegungsgründen für die Wahl dieser Sprachen bzw. Toolkits. Im Konkreten werden zwei aufeinander aufbauende Fragen verfolgt: Wieso wird in der gegebenen Abschlussarbeit auf eine Entwicklungsumgebung mit Python, Kivy und Plyer zurückgegriffen? Welche Vorteile hat es dabei, insbesondere Kivy als Framework für die UI-Gestaltung einzusetzen und wo treten Schwierigkeiten auf?

Unter den Quellen zur genannten Thematik sind kaum aktuelle Paper oder Untersuchungen zu finden. Deshalb dienen in erster Linie Lehrbücher und die offiziellen Dokumentationen als Wissensgrundlage. Im Falle der Programmiersprache Python ist hier beispielsweise das Werk von Grotz [2] zu nennen. In Bezug auf das Framework Kivy, geben Solis [6] und Vasilkov [8] brauchbare Einführungen und Beispiele in die Applikationsgestaltung. Weiterhin sind die offiziellen Softwaredokumentationen von Kivy [3] sowie Plyer [4] übersichtlich und verständlich aufgebaut, sodass problemlos darauf zurückgegriffen werden kann.

Im Folgenden werden zunächst Gründe dargelegt, die für die Verwendung von Python als Programmiersprache zur Implementierung von Logikbestandteilen sprechen. In Abschnitt zwei wird genauer auf Kivy als Graphik-Toolkit eingegangen. In diesem Zusammenhang werden Vorteile des Frameworks herausgearbeitet sowie die spezielle „Kivy language“ zur Beschreibung der UI-Elemente vorgestellt. Weiterhin wird auf Probleme eingegangen, die bei der App-Entwicklung mit Kivy auftreten. Der anschließende dritte Abschnitt beschäftigt sich mit Plyer als Schnittstelle zur Hardware, bevor in Abschnitt vier ein zusammenfassendes Fazit gezogen wird.

# Python

In der Arbeit wird Python eingesetzt, um Funktionalitäten, Algorithmen und dynamische Abläufe zu realisieren. Während der Implementierungsphase auf einem Windows-Betriebssystem wird Python in der Version 3.7.0 verwendet. Um die entstandene App auf das Endgerät zu übertragen, wird der Kivy Launcher verwendet (vgl. **Verweis einfügen**), auf dem die Programmiersprache in der Version 2.7.2 vorinstalliert ist. Deshalb ist bei der Implementierung der Anwendung darauf geachtet worden, dass sie sowohl für Python 2, als auch für Python 3 Kompatibilität besitzt.

Python wird als Programmiersprache im Rahmen des Moduls „Angewandte Informatik“ (vgl. [1, 17f]) im Studiengang Technische Physik gelehrt. Sie besitzt unter anderem den Vorteil, dass sie vergleichsweise einfach erlernt werden kann. Auf ihrer Basis ist es dementsprechend schnell möglich, Applikationen zu programmieren. Weiterhin besitzt Python-Syntax eine gute Lesbarkeit, wodurch es dem Benutzer zusätzlich erleichtert wird, die eigenen Ideen in ausführbaren Code umzusetzen. Neben diesen Vorteilen, existieren diverse Mathematik-Module in Python, weshalb die Sprache allen voran im akademischen und schulischen Bereich weit verbreitet ist [2, S. 3].

Die Software ist kostenlos und Open-Source, sodass sie frei zum Download zur Verfügung steht. Außerdem kann sie auf diversen Betriebssystemen verwendet werden. Somit ist es möglich, von beliebigen Computern auf die Dateien zuzugreifen, was eine flexible Entwicklung an unterschiedlichen Orten bzw. Geräten erlaubt [5, S. 18].

Es erscheint daher unter den gegebenen Rahmenbedingungen sinnvoll, Python für die Programmlogik heranzuziehen. Des Weiteren, da, zum Zeitpunkt der Anwendungserstellung, weder in Folge des Studiums, noch durch sonstige Ausbildungen, weitere Programmierkenntnisse in alternativen Sprachen vorliegen. Das Erlernen einer neuen Programmiersprache von Grund auf entzieht sich somit dem Rahmen einer Bachelorarbeit, zudem bereits die Einarbeitung in zusätzliches graphisches Framework notwendig ist, um die Benutzeroberfläche zu realisieren

# Kivy

Um den Programmcode übersichtlicher zu gestalten und die Logik von der graphischen Erscheinung trennen zu können, benötigt die Bachelorarbeit neben Python das Framework Kivy. Allerdings findet sich für die Entwicklung von Applikationen eine Fülle an guten Plattformen, Toolkits und Frameworks, wie Qt oder Flash [3], die eine Alternative zu Kivy bilden. Dieser Abschnitt beschäftigt sich daher genauer mit den Grundsätzen, den Vorteilen und der Architektur des verwendeten Toolkits. Zudem wird auf Schwierigkeiten eingegangen, die während der Entwicklung aufgetreten sind.

## Hintergründe und Motivation

Im Jahr 2011 wird Kivy von den Core-Entwicklern Virbel, Hansen und Lobunets auf der Tagung „Mensch und Computer 2011“ als Software Toolkit zur Entwicklung der „next generation of user interfaces“ [9] vorgestellt. Darin beschreiben sie ihre primäre Motivation für das Projekt: Den Mangel an plattformübergreifenden Werkzeugen, um native Benutzeroberflächen (NUI) entwickeln zu können.

Besonderen Wert legen sie bei ihrem Vorhaben auf zwei Schlüsselaspekte der NUI-Entwicklung, Verarbeitung von User-Input und Rendern des Outputs. Ersteres setzt ein optimiertes Reagieren auf Interaktionen mit dem Nutzer, wie z.B. Touch-Events voraus. Mit Hilfe neuer Event-Modelle können Hardware-Kapazitäten erweitert und zusätzliche Daten innerhalb des Touch-Events gespeichert werden. Letzteres benötigt eine performante, moderne Graphik-Hardware, die durch das OpenGL ES2 Framework gegeben wird. Alles in allem soll damit ein Entwickler-freundliches Framework geschaffen werden, welches plattformübergreifend auf Windows, Mac OSX, Linux und Android verwendet werden kann.

## Vorteile und Leistungen von Kivy

Kivy ist als Weiterentwicklung des PyMT-Projekts entstanden (vgl. [9]) und besticht im Vergleich zu anderen Frameworks durch entwicklerfreundliche Features, die im Folgenden detaillierter ausgeführt werden.

### Flexible und Plattformübergreifende Kompatibilität

Kivy erhebt den Anspruch, auf allen größeren Betriebssystemen laufen zu können. Dazu zählen neben Windows, Linux und OS X auch Android-Smartphones und Tablets [3]. Eine derartige plattformübergreifende Software wird heutzutage mehr und mehr zur Standardanforderung. Besonders vor dem Hintergrund der breiteren Zielgruppe und den daraus resultierenden größeren Verkaufszahlen, lohnt es sich in die Entwicklung der umfangreichen Kompatibilität zu investieren. Zudem wirkt sich eine problemlose Portierbarkeit und Verwendung von Applikationen positiv auf den Ruf der zugrunde liegenden Technologie aus [8, S. 1].

Mit der graphischen UI-Bibliothek Kivy wird somit in gewisser Weise das Motto „write once, run anywhere“ wieder aufgegriffen, welches sich die Programmiersprache Java bereits 1995 zum Ziel gesetzt hatte. Kivy verspricht nun diese Kompatibilität zwischen den Plattformen zu gewährleisten und Python-Anwendungen mit nativen Benutzeroberflächen auszustatten. Durch das Framework ist es möglich, eine Vielzahl unterschiedlicher User-Eingaben mit dem gleichen Code zu verarbeiten. Dies ermöglicht eine API-Gestaltung, die, nach außen hin, vereinheitlichte Schnittstellen bietet und intern auf bewährte Komponenten zurückgreift. So nutzt Kivy, je nach Bedarf, zahlreiche bekannte Bibliotheken, wie Pygame, SDL oder GStreamer. Dies wird nicht zuletzt durch den „Python Package Index“ (PyPI) realisiert, einer integrierten Standard-Bibliothek mit einer Fülle an Softwarepaketen von Drittanbietern [8, S. 2].

Die Flexibilität des Toolkits zeigt sich weiterhin nicht nur in Bezug auf die Unterstützung diverser Betriebssysteme, sondern auch in der Anpassung an neue Technologien. Kivy ermöglicht durch die rasche Weiterentwicklung stetig den Support von neuen externen Geräten und Software-Protokollen – im Idealfall bereits vor deren offiziellen Releases. Schließlich kann Kivy in Kombination mit einer Vielzahl an Drittanbieter-Lösungen zu genutzt werden. Beispiele hierfür sind die Unterstützung von „WM\_TOUCH“ auf Windows-Plattformen oder Apples Multi-Touch fähiger Input-Geräte. Derartige Funktionalitäten sind in Kivy ebenso verwendbar, wie sogenannte „HID kernel input events“ des Linux-Systems und zahlreiche weitere Eingabequellen [3].

### Entwicklerfreundlichkeit

Zur Entwicklung graphischer Oberflächen verknüpft Kivy seine eigenen Sprachkonstrukte mit der Codebasis aus Python. So integriert sich die speziell entwickelte *Kivy Language* (vgl. Abschnitt C) in eine mächtige, vielseitige und leicht zu erlernende Programmiersprache. Mit Hilfe der spezifischen, beschreibenden Syntax können Elemente innerhalb einer Applikation schnell und einfach initialisiert, verbunden und angeordnet werden. Programme in einer Entwicklungsumgebung von Python und Kivy aufzusetzen verspricht außerdem die Konzentration auf den Entwicklungsvorgang selbst. Es sind keine komplizierten und facettenreichen Einstellungen nötig, wie z.B. Compiler-Settings. So wird dem Programmierenden Aufwand abgenommen, der stattdessen in die Anwendung selbst gesteckt werden kann und dem Bearbeitungsprozess zu Gute kommt. Dadurch stellt Kivy eine Bereicherung in Punkto Entwicklerfreundlichkeit dar [3].

### Neuartigkeit des Frameworks

Neue Input-Methoden, allen voran „Multi-Touch“-Events bilden in modernen Benutzeroberflächen einen bedeutenden Faktor. Um die Interaktion zwischen Nutzer und Applikation so komfortabel wie möglich zu gestalten, haben die Entwickler mit Kivy ein vollkommen überarbeitetes Toolkit auf den Markt gebracht. So enthält die Software keine etablierten Architekturen oder Modelle, wie es in existierenden Lösungen häufig der Fall ist. Im Zuge der grundlegenden Neuimplementierung des Frameworks ist es stattdessen möglich, bekannte Schwierigkeiten bereits im Kern zu adressieren. Der Programmcode wird den modernen Bedürfnissen entsprechend aufgebaut, sodass der Computer nicht in festgelegte Strukturen gezwungen wird. Vielmehr analysiert Kivy die Interaktion zwischen Mensch und Maschine, um diese bestmöglich zu unterstützen und legt seine eigene Implementierung auf die Ansprüche moderner Technik aus [3].

### Schnelligkeit

Die Ausführungsgeschwindigkeit von Kivy auf mobilen Geräten ist vergleichbar mit den nativen Systemen, wie Java für Android oder Objective C für iOS. Darüber hinaus hat das Toolkit den Vorteil auf unterschiedlichen Betriebssystemen verwendet werden zu können, wie es viele Entwickler z.B. von HTML5 gewohnt sind. Im Gegensatz zu derartigen Technologien ist Kivy allerdings von keinen Zusatzprogrammen, wie z.B. Webbrowsern, abhängig. Dementsprechend weist das Framework eine bessere Performance auf. Viele seiner Komponenten sind in der Programmiersprache C implementiert, was durch die Cython-Bibliothek ermöglicht wird und die Kompilierung der Applikationen optimiert. Weiterhin läuft der Großteil der Graphikverarbeitung direkt in der GPU des Endgeräts ab [7, S. 1]. Da die Rechenleistung moderner Graphikkarten in Bezug auf Algorithmen, die die graphische Darstellung betreffen, derjenigen von CPUs überlegen ist, wird hierdurch die Performance beträchtlich erhöht [3]. Unter Verwendung der schnellen OpenGL Rendering Technologie wird so die Programmierung von graphisch anspruchsvollen Anwendungen realisiert. Dies ermöglicht, neben der Entwicklung von Videospielen, auch ein verbessertes Nutzererlebnis mit reibungslosen Darstellungsübergängen [8, S. 2].

1 Label:

2 text: 'Hello, Kivy'

Fig. 1. Modifizierung eines Labels in Kivy Language

### Professionelle Weiterentwicklung

Mit Kivy besteht ein Framework, welches professionell und zukunftsträchtig weiterentwickelt wird. Die Software wird zum Teil aktiv von Entwicklern vorangetrieben, die dieser Aufgabe hauptberuflich nachgehen. Zudem existiert eine breite Gemeinschaft rund um Kivy und die Vermarktung des Toolkits ist entsprechend organisiert [3].

### Open Source

Als Open-Source-Software ist Kivy MIT lizensiert und vollkommen kostenlos nutzbar. Der gesamte Sourcecode wird auf GitHub gehostet und ist frei einsehbar. Dementsprechend ist das Lösen von Software-Bugs und Hinzufügen neuer Features jedem erlaubt. Des Weiteren darf die Software kommerziell eingesetzt werden, ohne dass Lizenzgebühren anfallen. [8, S. 2]. Somit ist anzumerken, dass Kivy nicht nur im privaten Umfeld frei verwendet, sondern auch Gewinn damit erzielt werden darf. Das bedeutet konkret, dass ein Entwickler Geld damit verdienen kann, indem er Applikationen verkauft, die er mit Kivy erstellt hat. Auch in diesem Fall bleibt die Benutzung des Toolkits für ihn kostenlos [3].

## Kivy Language

Kivy verwendet eine eigene Sprache, um User Interfaces einfach und anwendungsspezifisch zu gestalten. Die sogenannte *Kivy Language* oder *Kv Language* trennt das Oberflächendesign von der Applikationslogik. Dies geschieht derartig, dass die Logik im Python-Code verbleibt, während die Gestaltung der Elemente in die Designsprache (in zugehörige *.kv*-Dateien) ausgelagert wird [6, S. 9].

Die Grundlage für die UI-Gestaltung in Kivy bilden Widgets. Jede Applikation besteht aus einem sogenannten *root widget* und hat in der Regel weitere Kindelemente. So wird ein Baum aus Widgets aufgebaut, die auf dem Bildschirm erscheinen, Events empfangen und darauf reagieren [3]. Wie Fig. 1 zeigt, können solche Widgets der Applikation über die *Kivy Language* intuitiv mit Eigenschaften versehen werden. In diesem Fall wird dem Label-Objekt der Text „Hello, Kivy“ hinzugefügt. Label ist dabei ein mitgeliefertes Widget, welches nach der Installation von Kivy zur Verfügung steht. Daneben können weitere Widget-Klassen definiert und bei Bedarf mit Standardwerten versehen werden. In analoger Weise lassen sich daraus anschließend Objekte ableiten, die zur Gestaltung der Oberfläche herangezogen werden.

Bei der *Kivy Language* handelt es sich somit um eine domänenspezifische Sprache (DSL), die den Zweck verfolgt die Elemente einer Applikation mit den gewünschten Eigenschaften auszustatten (vgl. [8, S. 13]). Dies betrifft das Layout der Widgets, genauso wie das Styling und die Erstellung von Klassen bzw. Unterklassen bestehender Komponenten. Weiterhin lassen sich Event-Handler-Funktionen direkt in den *.kv*-Files angeben, sodass die Widgets auf Ereignisse reagieren können. Die entsprechende Ausimplementierung der Funktionalitäten erfolgt im korrespondierenden Python-Code, wodurch die Trennung zwischen graphischer Erscheinung und Applikationslogik erhalten bleibt.

## Applikationsentwicklung auf Android-Endgeräten

Im Rahmen der Abschlussarbeit soll die Applikation auf einem Tablet (Asus ZenPad 10) lauffähig sein, welches das Betriebssystem Android (Version 7.0) verwendet. Laut der offiziellen Dokumentation (vgl. [3]) gibt es mehrere Möglichkeiten, Anwendungen auf Android-Endgeräten lauffähig zu machen. In der Folge werden das „Buildozer“-Tool und der „Kivy Launcher“ vorgestellt. Ersterer Ansatz kann in der Arbeit nicht erfolgreich umgesetzt werden, weshalb schließlich auf Letzteren zurückgegriffen wird.

### Buildozer

Buildozer ist ein Automatisierungswerkzeug für das „python-for-android“-Projekt und bildet laut Kivys Dokumentation [3] den einfachsten Weg, um ein APK (Android Package Kit) zu erstellen. Diese Installationsdatei kann anschließend direkt auf das Endgerät übertragen werden und ist dort lauffähig. Da für die Applikationsentwicklung ein Windows-Rechner verwendet wird, das Tool aber zur Zeit der Entwicklung ausschließlich mit dem Linux-Betriebssystem kompatibel ist, wird auf eine Virtuelle Maschine zurückgegriffen. Wie in der Abschlussarbeit [10] detailierter ausgeführt, gelingt es darüber ein sogenanntes „buildozer-spec“-File zu erzeugen, auf Basis dessen die Anwendung erzeugt werden soll. Trotzdem schlägt die endgültige Generierung fehl und das App-Deployments bleibt in dieser Form erfolglos.

### Kivy Launcher

Auf Grund angeführter Probleme mit der Applikationserstellung über das Buildozer-Tool, wird im Zuge der Bachelorarbeit auf den Kivy Launcher zurückgegriffen. Der Launcher macht es möglich, Kivy-Apps auf Android-Endgeräten zu starten, ohne eine vorhergehende Kompilierung notwendig zu machen (vgl. [3]). Voraussetzung hierfür ist der Download des Programms über den Google Play Store. Zur Zeit der Ausarbeitung ist der Launcher in der Kivy-Version 1.9.1 kostenlos erhältlich. Damit eine selbst implementierte Anwendung auf dem Android-Endgerät funktioniert, muss sie schließlich an der korrespondierenden Speicherstelle und dem Pfad */sdcard/kivy/<yourapplication>* abgelegt werden. Dabei steht der Platzhalter „<yourapplication>“ für ein Verzeichnis, in dem wenigstens die Datei main.py und eine Textdatei *android.txt* zu finden sind. Weitere Dateien der Anwendung, z.B. *.kv-*Files, Bilder oder Ähnliches, können ebenfalls in diesem Verzeichnis abgelegt werden. Schließlich sollte das *android.txt* Textfile Informationen enthalten, die über Autor, Titel und Orientierung des Bildschirms Auskunft geben. Das entsprechende File für die Science-Escape-Box-App ist in Fig. 1 abgebildet. Damit lässt sich die Anwendung schließlich über den Launcher öffnen.



Fig. 2. android.txt Datei für die Science-Escape-Box Applikation

## Some Common Mistakes

* The word “data” is plural, not singular.
* The subscript for the permeability of vacuum **0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter “o”.
* In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
* A graph within a graph is an “inset”, not an “insert”. The word alternatively is preferred to the word “alternately” (unless you really mean something that alternates).
* Do not use the word “essentially” to mean “approximately” or “effectively”.
* In your paper title, if the words “that uses” can accurately replace the word “using”, capitalize the “u”; if not, keep using lower-cased.
* Be aware of the different meanings of the homophones “affect” and “effect”, “complement” and “compliment”, “discreet” and “discrete”, “principal” and “principle”.
* Do not confuse “imply” and “infer”.
* The prefix “non” is not a word; it should be joined to the word it modifies, usually without a hyphen.
* There is no period after the “et” in the Latin abbreviation “et al.”.
* The abbreviation “i.e.” means “that is”, and the abbreviation “e.g.” means “for example”.

An excellent style manual for science writers is [7].

# Plyer

* Kivy Documentation, S. 18

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

**The template is designed for, but not limited to, six authors.** A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

### For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than six authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

#### Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

# Zusammenfassung

##### Acknowledgment *(Heading 5)*

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

##### References

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first ...”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

1. G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
3. I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
4. K. Elissa, “Title of paper if known,” unpublished.
5. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
6. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
7. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord “Format” pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

Literatur

[1] „Bachelorstudiengang „Technische Physik“: Studienrichtungen „Physikalische Technologien“ und „Laser- und Lichttechnologie““, 2020. [Online]. Verfügbar unter: https://www.hs-coburg.de/fileadmin/hscoburg/Dokumente\_Studium/Modulhandbuch\_Bachelor\_Technische\_Physik.pdf

[2] B. Grotz, *Grundkurs Python 3: Version 0.1.2c*. Augsburg, 2017.

[3] Kivy, *Kivy Dokumentation: Release 2.1.0.dev0.* Verfügbar unter: www.kivy.org. Zugriff am: 4. Januar 2021.

[4] Mathieu Virbel, Akshay Aurora, Gabriel Petier und Ben Rousch, *Plyer Documentation.* [Online]. Verfügbar unter: https://plyer.readthedocs.io/\_/downloads/en/latest/pdf/ (Zugriff am: 4. Januar 2021).

[5] W. D. Sande und C. Sande, *Hello World!: Programmieren für Kids und andere Anfänger/ Warren D. Sande ; Carter Sande,* 2. Aufl. München: Hanser, 2014. [Online]. Verfügbar unter: http://www.hanser-elibrary.com/isbn/9783446438149

[6] H. Solis, *Kivy Cookbook*. s.l.: Packt Publishing, 2015. [Online]. Verfügbar unter: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&AN=1056297

[7] R. Ulloa, *Kivy - Interactive Applications and Games in Python: Second Edition*. Create responsive cross-platform UI/UX applications and games in Python using the open source Kivy library, 2. Aufl. Birmingham: Packt Publishing, 2015.

[8] M. Vasilkov, *Kivy Blueprints*. Birmingham: Packt Publishing, 2015. [Online]. Verfügbar unter: http://gbv.eblib.com/patron/FullRecord.aspx?p=1935723

[9] M. Virbel, T. Hansen und O. Lobunets, „Kivy – A Framework for Rapid Creation of Innovative User Interfaces“ in *Workshop-Proceedings der Tagung Mensch & Computer 2011*, M. Eibl und M. Ritter, Hg., 2011, S. 69–74.

[10] A. Wolfschmitt, „Entwicklung einer plattformübergreifenden App in Python zur Digitalisierung eines naturwissenschaftlichen Escape Games“, Hochschule für angewandte Wissenschaften Coburg, Coburg, 2019. Zugriff am: 4. Januar 2021.