Switch Mode 建立視覺化程式語言和文字式程式語言的中間過渡地帶
Switch Mode: Building a Middle Ground between Block-based and Text-based Programming

林榆涵 [1]， David Weintrop [2], Jason McKenna [3] , 羅木迪 [4]
Yuhan Lin[1], David Weintrop [2], Jason McKenna [3], Mudi Luo [4]

[1] 馬里蘭大學 教育與學習暨政策與領導力研究 博士生
[1] University of Maryland Department of Teaching and Learning, Policy and Leadership Doctoral Student
E-mail: jimmylin@umd.edu
[2] 馬里蘭大學 教育與學習暨政策與領導力研究 教授
[2] University of Maryland Department of Teaching and Learning, Policy and Leadership Assistant Professor
E-mail: weintrop@umd.edu
[3] VEX 機器人公司 全球教育策略總監
[3] VEX Robotics Director of Global Educational Strategy
E-mail: jason@vex.com
[4] 淡江大學 教育科技系 碩士生
[4] Tamkang University Department of Educational Technology Master Student
E-mail: 607734018@gms.tku.edu.tw

摘要

本研究介紹了一款設計策略 Switch Mode，它為學習者從視覺化程式語言轉向文字式程式語言提供了一個中間過渡地帶。Switch Mode 策略讓學習者在視覺化程式積木裡面編輯文字程式。Switch Mode 的添加方式是透過從程式積木區將 Switch Mode 積木拖放到程式編輯區中（圖 1(a)）或透過右鍵點擊傳統程式積木並選擇轉換選項（圖 1(b)）。Switch Mode 透過提供一種支援學習者在視覺化程式設計環境中引入文字式程式設計的方式，可以滿足使用者在保有當前視覺化程式設計能力水平情況下，同時幫助他們過渡到更複雜且強大的程式語言。Switch Mode 透過為學習者提供在具有視覺化程式設計工具支援的情況下嘗試文字式程式設計的方法，為了引導初學者進入文字式程式設計之不斷增長的工具生態系統做出了新的貢獻。

關鍵字： 程式語言的環境設計、教育機器人、視覺化程式設計、Switch Mode、文字式程式設計

Abstract

This paper introduces Switch mode, a design strategy to introduce a middle ground to support learners in transitioning from block-based to text-based programming. The

Switch mode strategy allows learners to author text-based commands inside a block-based programs. Switch mode blocks can be added by directly dragging and dropping Switch mode blocks from the block's pallet to the program canvas (Figure 1a) or by right clicking a conventional block and selecting the convert option (Figure 1b). In embedding a scaffolded way to introduce text-based programming in a blocks-based environment, Switch mode can meet the user at their current level of ability in blocks, while also help them move to more sophisticated and powerful programming languages. In providing a means for learns to tinker with text-based programming while still having the supports of block-based tools, Switch mode contributes a novel addition to the growing ecosystem of tools designed to transit novices to the practice of text-based programming.

Keywords: Design of Programming Environments, Educational Robotics, Block-based programming, Switch mode, Text-based programming

壹、緒論

(a) (b) (c)

圖 1：Switch Mode 介面和 Switch Mode 加入程式的兩種方法：(a)拖放 Switch Mode 積木，(b)將傳統程式積木轉為 Switch Mode 積木 和 (c) VEX VR

視覺化程式設計是一種向年輕學習者介紹程式設計以及更廣泛的電腦科學領域的方式 (Flannery et al., 2013; Maloney et al., 2010; Resnick et al., 2009)。隨著針對各年齡層設計的視覺化程式設計環境產生，最小可從幼稚園開始學習程式設計(Bers, 2018; Flannery et al., 2013)。視覺化程式設計是透過使用視覺化提示表示指令該如何配合使用以及禁止將不兼容的指令拼接在一起來減少語法錯誤(Maloney et al., 2010)。然而，對於有興趣進一步學習電腦科學領域或將來職業需要依賴程式設計的學習者而言，視覺化程式設計只是學習文字式程式設計的起點。隨著學習者進入高中或更高學歷，他們將需要從視覺化程式設計轉向傳統的文字式程式語言，例如 Python 或 Java。當學習者從視覺化程式設計離開轉向文字式程式設計的過程不是一帆風順，因為他們需要面臨額外的步驟及概念的轉變 (Kölling et al., 2015)。

雖然從視覺化程式設計過渡到文字式程式設計可能會有挑戰 (Weintrop & Wilensky, 2019)，但目前已經有小部分資源可用於幫助學習者面對此轉換的過程。幫助學習者從視覺化程式設計過渡到文字式程式設計，需要幫助學習者認識兩種程式設計模式之間的概念一致性，將在視覺化程式設計中編寫程式方式與文字式程式設計仍然有效的編寫程式方式建立連結，同時發展文字式程式設計上的新技能與程式編寫方式。除了在編寫程式模式上的轉變以外，兩種程式設計的程式除錯方式也存在差異，因為每種程式設計模式都有不同的特點和工具來幫助解決程式所產生的問題。我們專為幫助學習者從視覺化程式設計模式過渡到文字式程式設計模式設計了 Switch Mode，一種將視覺化程式設計與文字式程式設計特性融合在虛擬機器人平台上的程式設計方式。本研究主要介紹 Switch Mode，並討論它關鍵的設計特點以及背後的原理。

貳、SWITCH MODE 設計

一、VEX VR——Switch Mode 平台的根基

VEXcode VR 是一個教育機器人程式設計環境，包括視覺化程式設計以及虛擬機器人環境(圖 1c)。VEXcode VR 包括一個完整的視覺化程式設計語言，用於控制虛擬機器人在名為遊樂園的虛擬環境中進行探索。此虛擬機器與 VEX 開發的實體機器人系統相匹配，但完全虛擬的實現意味著學習者不必擁有實體機器人。

VEXcode VR 是基於 VEXcode 程式語言設計的，該程式語言設計借鑑了 Scratch Blocks 的介面。除了可以使用視覺化程式設計控制虛擬機器人的外，也可以讓學習者查看他們編寫的視覺化程式語言如何用 Python 實現。學習者還能夠將他們的視覺化程式導出成 Python，從而可以在文字式程式設計環境中繼續編寫。考慮到 VEXcode VR 存在從視覺化程式設計過渡到文字式程式設計的轉換路徑，因此認為 VEXcode VR 很適合引入 Switch Mode 功能，該功能提供文字式程式設計初體驗的機會，同時也可以繼續使用視覺化程式設計的工具。

二、Switch Mode 和 Switch Mode 積木設計

Switch Mode 是一種新型視覺化程式設計功能，讓學習者在視覺化程式積木中編寫一行或多行文字式程式。Switch Mode 的設計目的在於幫助初學者編寫文字式程式，同時保留視覺化程式設計的輔助工具。Switch Mode 積木可以像一般視覺化程式積木一樣使用，但它們並非用於指定特定的行為，而是在其中嵌入一個文字式程式編輯器，提供學習者輸入指令。此外，學習者可以透過右鍵點擊並選擇轉換選項來將一般的程式積木轉換為 Switch 積木。事先定義好的 Switch 積木支持單行指令、多行程式、變數、巢狀結構、函數積木。

(a) (b)

圖 2:(a)Switch Mode 在視覺化程式中的範例(b)Switch Mode 積木類型範例

三、除錯功能設計

Switch Mode 不僅支持在視覺化程式設計介面進行文字式程式設計，也包括了一系列支持學習者程式除錯的功能。

(一) 監控變數

學習者可以透過兩種方式來監控變數或是傳感器的數值。在一般視覺化程式設計環境中，學習者可以透過(1)拖放傳感器或變數積木到監控台圖標 (2)使用 Python 指令監控感應器或變數。對於第一種方式，學習者也可以從左側程式積木區拖放傳感器或變數積木到監控台圖標（見圖 3）.學習者也可以從中間的編輯區拖放積木到監控台圖標，積木會返回原本的積木區，不會干擾現有的程式。在 VEXcode 介面的右側，監控台螢幕將顯示被監控的變數及其數值。如果學習者拖動現有傳感器，該行將會突出顯示並閃爍 3 秒。另一種監控變數的方式是使用 Switch Mode 積木，並用 Python 輸入指令，此功能展現了 Switch Mode 積木可以擴充程式積木的能力以及其靈活性。使用 `monitor_variable("")` 或 `monitor_sensor("")`指令，在雙引號中加入要監控的變數，將變數加入監控列表中，以便在程式運行期間進行監控。

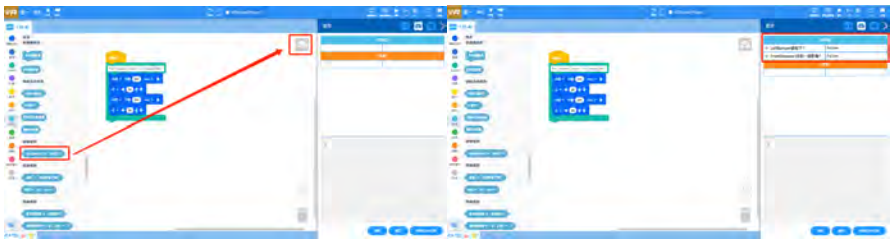

圖 3:拖放感應器程式積木至監控變數

(二) 錯誤信息

初學者在學習程式設計時面臨一個重大的挑戰——解釋錯誤信息(Nienaltowski et al., 2008)。其中包括識別錯誤定位和錯誤類型。為了更好幫助學習者解釋錯誤信息並除錯，我們設計了三種不同的錯誤信息顯示方式。第一種顯示方式是在 Switch Mode 積木的左側顯示一個帶有警告圖標的提示框幫助學習者識別該積木中存在錯誤。圖標會彈出一個對話框，顯示簡化和容易閱讀的錯誤提示。系統還可以針對一些錯誤提供建議的解決方案。圖 4 的範例中，學習者在指令中缺少一個逗號，錯誤提示顯示“Syntax Error: Please check for missing comma, parentheses, dot.”，這種顯示簡化錯誤提示的方法可以幫助學習者定位錯誤以及如何除錯。儘管該錯誤提示沒有提供明確的建議解決方案，但它為除錯過程提供支持與幫助，幫助學習者建立定位、識別和解決程式錯誤所需要的基礎技能。此方法也突出了 Switch Mode 方法的一個特色，因為程式只有少數幾個文字化程式的指令，這表示錯誤來自這組指令。例如圖 4b 中，只有一個語法錯誤可能的來源，幫助學習者清楚地定位錯誤的位置，而不需要解析整段程式的五行指令。

(a) (b) (c) (d)

圖 4:(a)標準 Switch 積木(b)含錯誤提示的 Switch 積木(c)螢幕底部的錯誤面板 (d)Python 指令查看器的錯誤提示

VEXcode 中顯示錯誤提示的第二個位置時在螢幕底部的錯誤面板（如圖 4c 所示），它顯示了出現錯誤的檔案。在此範例中，只有一個檔案，所以它顯示的是“viewer_main.py”檔案。然後它的第一行顯示了“Syntax Error (4,44)“，這是 Python 錯誤提示的標準顯示方式，告知學習者造成錯誤的指令所在的行和列號。然而大多數學習者不清楚(4,44)在指令查看器的上下文中意味著什麼。為提供更多訊息給學習者，我們提供了兩行提示“- Please check for missing commas, parentheses, and dots. - Please check code viewer line 4。”因

此它幫助學習者辨識出指令查看器中的第四行。未來我們規劃在指令查看器的左側提供帶有錯誤圖標的第四行第 44 字符的提示。提供原始未解析的錯誤提示的目的是為學習者邁向傳統 Python 編輯器中編寫程式做準備。

第三種錯誤提示顯示方式是在 Python 指令查看器中（圖 4d）。錯誤的一行被黃色方框框起，左側也有一個錯誤標誌。除了框起的錯誤以外，監控台下方的終端面板也會有完整的追蹤日誌，類似 Python 錯誤提示的標準顯示方式。因此學習者可以在完整的 Python 程式旁邊查看最原始的錯誤日誌。再次強調，此設計的目的是為了讓學習者預備脫離 VEXcode 邁向 Python。

總結以上三種針對各個階段的學習者提供的錯誤提示顯示方式，包括錯誤提示框、錯誤面板、終端視圖。

表 1: 多種錯誤提示的範例

Error Bubble	Error Panel	Terminal View
Syntax Error: Please check for missing comma, parentheses, dot.	Syntax Error (4,44) - Please check for missing commas, parentheses, and dots. - Please check code viewer line 4.	Traceback (most recent call last): File "<exec>", line 8, in <module> File "<exec>", line 25 await drivetrain.drive_forward(FORWARD, 200 MM) ^ SyntaxError: invalid syntax
NameError: Found a variable/function that does not exist in the code -	NameError: Found a variable/function that does not exist in the code – specified variable/function/class name does not exist	Traceback (most recent call last): File "/lib/python3.8/asyncio/tasks.py", line 280, in __step result = coro.send(None) line 32, in when_started1 NameError: name 'mm' is not defined

參、SWITCH MODE 與領域的研究貢獻

本研究建立在大量研究的基礎上，探討設計出支持初學者程式設計到專業程式設計的程式設計環境。基於 Switch Mode 的設計目標是從視覺化程式設計過渡到文字式程式設計，其設計靈感來源於各種關注視覺化程式設計的研究 (Brown et al., 2021; Lin & Weintrop, 2021; Weintrop & Wilensky, 2015)。Switch Mode 探索出一種銜接兩種程式設計形式之間差距的潛在辦法，對程式設計教育以及數位學習領域做出新的貢獻。

參考文獻

Bers, M. U. (2018). Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 2094–2102. https://doi.org/10.1109/EDUCON.2018.8363498

Brown, N., Kyfonidis, C., Weill-Tessier, P., Becker, B., Dillane, J., & Kölling, M. (2021). A Frame of Mind: Frame-based vs. Text-based Editing. *United Kingdom and Ireland Computing Education Research Conference.*, 1–7. https://doi.org/10.1145/3481282.3481286

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *Proceedings of the 12th International Conference on Interaction Design and Children*, 1–10. https://doi.org/10.1145/2485760.2485785

Kölling, M., Brown, N. C. C., & Altadmri, A. (2015). Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*, 29–38. https://doi.org/10.1145/2818314.2818331

Lin, Y., & Weintrop, D. (2021). The landscape of Block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming. *Journal of Computer Languages*, 101075. https://doi.org/10.1016/j.cola.2021.101075

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 1–15. https://doi.org/10.1145/1868358.1868363

Nienaltowski, M.-H., Pedroni, M., & Meyer, B. (2008). Compiler error messages: What can help novices? *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 168–172. https://doi.org/10.1145/1352135.1352192

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*, 199–208. https://doi.org/10.1145/2771839.2771860

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, *142*, 103646. https://doi.org/10.1016/j.compedu.2019.103646