International Society of the Learning Sciences

Programming Time: Exploring Time as a Cultural Construct Across Novice Computational Platforms

Ravi Sinha, Phyllis Kyei Mensah, Breanne K. Litts
ravi.sinha@usu.edu, phyllis.kyeimensah@usu.edu, breanne.litts@usu.edu
Utah State University

Rogelio E. Cardona-Rivera, University of Utah, rogelio@eae.utah.edu
Melissa Tehee, Utah State University, melissa.tehee@usu.edu

Abstract: Public critiques of technologies and the algorithms that power them have pushed designers to critically consider for whom they design and who they include in design processes. In education, similar critiques highlight how computational technologies designed for novice learners commonly privilege certain ways of knowing and being. In response, this poster explores how the cultural construct of time is represented across computational platforms for novices and what this means, particularly for Indigenous learners and designers.

Introduction

Issues of (mis)representation and lack of representation of minoritized groups have shaped the underlying computational models that drive technology "innovation." This has caused profound harm to these groups, particularly Indigenous communities. Given that industry leaders in technology design, such as Apple, Google, and Microsoft, cannot resolve this misrepresentation (Eubanks, 2018; Noble, 2018; O'Neil, 2016), education scholars also find similar issues in education technologies (Litts et al., 2021). The solution requires acknowledging how these technologies and novice computational platforms are themselves instantiations of cultural systems that privilege particular ways of knowing and being in the world.

As an example of how misrepresentation is embedded in novice computational platforms, scholars have found that Indigenous cultures tend to use event-based time intervals that rely on events, seasons, natural cycles and elements, social norms, etc. (Sinha, 2019), rather than the quantified time-based metric interval systems such as clocks, seconds, minutes and hours (Sinha et al., 2011), which many novice computational platforms tend to rely on. In response to this persistent need for novice computational platforms to design for diverse interpretations and representations of time, we investigate the research question: *how do novice computational programming platforms represent the cultural construct of time*? To address this question, we analyzed how time is represented in 45 computational platforms designed for novice learners. Our analysis illustrates how these platforms can privilege particular cultural ways of knowing and being. Findings highlight opportunities to design for accessible forms of programming that align with Indigenous representations of time.

Methods

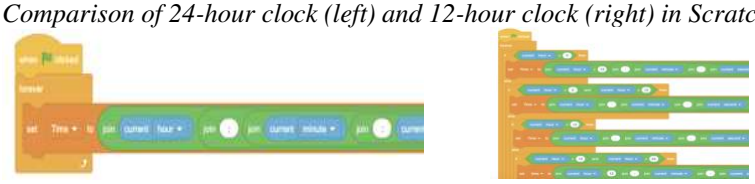
We curated a list of 45 novice computational platforms that are frequently used in K-12 settings using search terms such as "block-based computational platforms," "digital storytelling platform," and "immersive storytelling platforms." Across platforms, we searched for representations of "time" as either a programming element on the platform or in tutorial blogs on the platform's online community. We identified time features in 29 of the 45 platforms and conducted further analysis to identify the specific ways in which time is represented.

Findings: Representing time

For this poster, we share how time is represented in three platforms: Scratch, Alice, and MIT App Inventor. These serve as illustrative examples of how time is represented across platforms. We found that there are accessible and simple ways to represent time functionality in a game or story. For example, in Scratch, the "current ()" block can be used to report the year, month, date, day of the week, hour, minute, or second in a project. The block displays time in a 24-hr format, and a date based on the device's local time. This affords learners to include a 24-hr clock and a Gregorian calendar date in their projects. However, it becomes more difficult to include date-time in other formats. For example, adding a 12-hour clock requires more complex coding (see Figure 1).

Figure 1

Comparison of 24-hour clock (left) and 12-hour clock (right) in Scratch.



In Alice, time is represented in projects using the event-listener option. For example, the event listener "addTimeListener," can execute actions after a certain time has elapsed. In the code snippet below (Figure 2a), after myFirstMethod is executed, the "walking footsteps" audio file plays after a delay of 0.25 seconds. In Figure 2b, to implement an animation where an eagle flies to a log and sits on it, the duration of each move - forward, up, down during flight can be controlled via the "delay-duration" option to further smoothen the visual effect in the scene.

Figure 2

Time in Alice. (left to right) (a) addTimeListener (b) Animating Eagle's flight

MIT App Inventor also provides learners a range of approaches to integrate time in their games and stories. Objects can be moved or transformed on canvas by setting the time intervals. Timer event is the most general method to define those set time intervals. Objects' properties can also be used to define those intervals. For instance, one can specify the "TimerInverval" property to control the animation effect. When moving the object, the smaller the interval, the faster the object will appear moving. The interval is defined in terms of milliseconds. Additionally, it also allows users to add a specific amount of time (e.g., hours, days, years, etc.).

Discussion

The novice computing platforms we examined provided accessible support to facilitate the representation of time in quantified time-metric intervals (seconds, minutes, hours) and dates using months and years after the Gregorian calendar. However, the platforms we explored do not inherently or explicitly provide structures and blocks that allow users to represent time with events, seasons, life, natural, and cosmic cycles, which are fundamental to Indigenous cultures and stories (Sinha, 2019). We envision a future where these platforms provide accessible structures and representations to easily integrate real-world elements and connections by, for example, drawing on weather or location data. While experienced users may rely on advanced skills to replicate event-based time intervals on these platforms, the existing underlying biases in the structures and representations pose significant limitations for novice users. Insights from our analysis reifies the argument made by scholars (e.g., Litts et al., 2021) to (re)examine the deeper design structures of the computational platforms and the implicit biases in their design toward the goal of designing culturally sustaining/revitalizing computational tools for all.

References

Eubanks, V. (2018). Automating inequality: How high-tech tools profile, police, and punish the poor. St. Martin's Publishing Group.

Litts, B. K., Searle, K. A., Brayboy, B. M., & Kafai, Y. B. (2021). Computing for all?: Examining critical biases in computational tools for learning. British Journal of Educational Technology, 52(2), 842-857.

Noble, S. U. (2018). Algorithms of oppression: How search engines reinforce racism. NYU Press.

O'Neil, C. (2016). Weapons of math destruction: How big data increases inequality and threatens democracy. Crown.

Sinha, C., Sinha, V. D. S., Zinken, J., & Sampaio, W. (2011). When time is not space: The social and linguistic construction of time intervals and temporal event relations in an Amazonian culture. Language and Cognition, 3(1), 137-169.

Sinha, V. D. S. (2019). Event-based time in three indigenous Amazonian and Xinguan cultures and languages. Frontiers in Psychology, 10. 1-24. https://doi.org/10.3389/fpsyg.2019.00454

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2119573 & 2119630. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.