Velo: Exploring Animal Behavior Modeling through Hybrid Robotics-Simulation Learning Experience

Kritphong Mongkhonvanit*
Tyler Hummer*
John Chen
kritphong@u.northwestern.edu
tylerhummer2027@u.northwestern.edu
civitas@u.northwestern.edu
Northwestern University
Evanston, Illinois, USA

ABSTRACT

Velo is a learning experience that combines robotics and simulation to help learners understand and apply a simple yet powerful programming model inspired by Braitenberg vehicles. In this model, programs are constructed only by making connections between *sensors* and *actuators*. Despite this simplicity, it is possible to achieve complex behaviors similar to that of animals. Velo is designed to be used in a curriculum that aims to help learners not only learn this programming model, but also the process of analyzing an existing (animal) behavior and breaking it down into a form useful for programming.

CCS CONCEPTS

• **Computer systems organization** → Robotics; • **Applied computing** → **Interactive learning environments**; **Computer-assisted instruction**; • **Software and its engineering** → *Visual languages*; Domain specific languages.

KEYWORDS

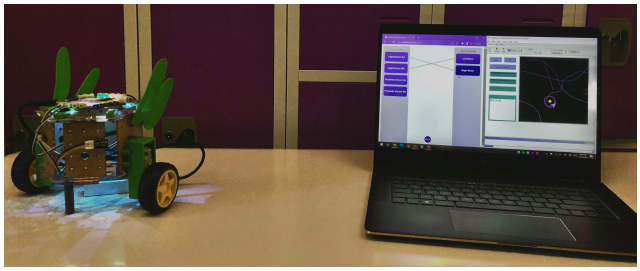
Braitenberg vehicles, physical computing, computer science education, visual programming language, learning, education

ACM Reference Format:
Kritphong Mongkhonvanit, Tyler Hummer, and John Chen. 2023. Velo: Exploring Animal Behavior Modeling through Hybrid Robotics-Simulation Learning Experience. In *Interaction Design and Children (IDC '23), June 19–23, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3585088.3594489

1 INTRODUCTION

Children like animals. According to Kellert [6], children tend to have *humanistic* attitudes towards animals, meaning that they have "primary interest and strong affection for individual animals". Moreover, children's level of knowledge about animals were found to be comparable to, or even exceed, that of adults in many areas. Not only can this existing knowledge and interest serve as an excellent basis for further learning about animals, it can also potentially, with appropriate scaffolding, serve as a starting point to learn about other topics.

We present Velo, a learning experience that combines robotics and simulation to help bridge children's interest in animals with computer programming and robotics. It combines robotics and simulation to help learners understand and apply a simple, yet powerful, programming model inspired by Braitenberg vehicles [1]. In this model, programs are constructed only by making connections between *sensors* and *actuators*. Despite this simplicity, it is possible to achieve behaviors with surprising levels of complexity similar to that of animals. Velo is designed to be used in a curriculum that aims to help learners not only learn this programming model, but also the process of analyzing existing behaviors and breaking it down into a form useful for programming.



2 DESIGN GOALS

Our goal is to create a learning experience that helps children aged 13–16 learn how to analyze animal behaviors using the model of Braitenberg vehicles, and how to apply that knowledge to build their own computer models. Velo is intended to be used as a part of a multi-modal, simulation, and robotics based curriculum that encourages learners to bridge their knowledge about animals and their behaviors with robotics and computer programming. The curriculum includes activities that guide students through the process of observing animal behaviors, breaking them down into simple rules, and finally encoding those rules in a program. Constructionist design principles [9] are used as the foundation for students to build upon the idea that even simple sensory inputs, along with equally simple neural connections, can lead to seemingly complex

*Both authors contributed equally to this research.

IDC '23, June 19–23, 2023, Chicago, IL, USA

ACM ISBN 979-8-4007-0131-3/23/06.
https://doi.org/10.1145/3585088.3594489

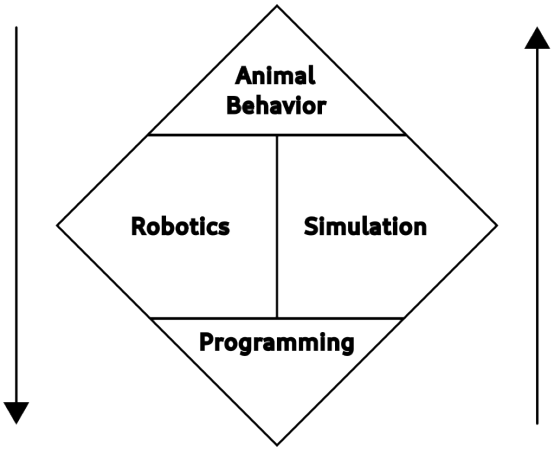


Figure 1: The overall conceptual structure of the system.

behaviors. Our simulator allows for these neural connections to be created in an easily digestible manner. The resulting behavior can then be studied within a customized environment with relevant stimuli. Further, we provide a robotics platform that can be used to study the same sensor-actuator neural connections used within the simulation in a physical robot. Keeping consistent with constructionist principles, this robot is designed as a platform to be used by learners for exploration, and can change in both form and function.

3 RELATED WORK

3.1 Braitenberg Vehicles

Braitenberg vehicles are robots designed to *sense*, *think*, and *act* using some of the most simple sensor-actuator neural connections possible. Despite the simplicity of the robot, complex behaviors similar to living organisms begin to emerge. Take, for example, the simple sensor-actuator neural configuration shown in Fig. 2 with a positive linear correlation between light sensors and wheels. As each sensor is exposed to more light, the speed of the wheel on the same side increases, thus sending the vehicle away from the light source. This behavior could be likened to that of a cockroach, where when a light switch is turned on, the roach being scared of the light, tries to run away as fast as possible. By changing the sensor-actuator correlations to include functions such as thresholds or non-linear relations, the types of animal behaviors that can be effectively mimicked and explored by students greatly increases.

3.2 Robotics and Simulation in Education

Robotics and simulation are powerful technological tools that, when utilized in educational settings, provide effective mediums for studying common or advanced topics [15]. With the tangibility and concreteness of robotics platforms, children as young as kindergarten age have been found to develop basic understandings of topics such as emergent behavior and general systems control [7]. Similarly, computer simulations have provided a platform for restructuring the methods with which advanced topics such as Newtonian physics and statistical mechanics are taught [16]. With the introduction of robotic platforms such as Cubelets [11], and agent-based simulation

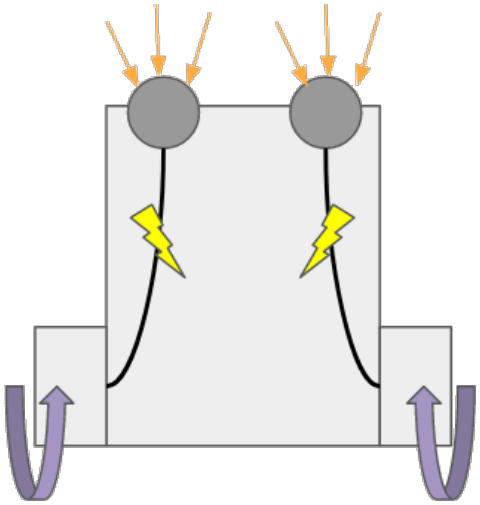


Figure 2: A simple Braitenberg vehicle with *direct* connections between sensors and motors on the same side.

programs like NetLogo[14], robotics and simulation platforms for use within educational settings are more accessible than ever. This accessibility allows experiences, like Velo, to explore new subject areas such as animal behavior.

3.3 Domain-specific Visual Programming

Visual programming languages allow users to construct programs by manipulating graphical representations of programming constructs. This approach can help overcome novice programmers' difficulties with syntax, allowing them to focus on the logic and structure involved in programming [2]. Examples of visual programming languages include Google Blockly[3], Scratch[10], NetTango Web[4], and DeltaTick[17], among others. Visual programming languages have been successfully used and found to be effective in educational contexts[13].

Domain-specific visual programming presents a novel way of designing programming environments for novices. For example, NetTango Web and DeltaTick allow complex systems modelers to design blocks for a specific domain (such as ants foraging or virus spreading) without invoking additional cognitive load of programming concepts[8, 17]. Such platforms encourage designers and learners to focus on the micro-level behaviors of individual components in complex systems, the domain-specific knowledge, instead of introductory computer science concepts. Consequently, children in museums were found capable to program frogs' behaviors within the first 3 minutes of interaction and engage in complex systems thinking [5].

4 DESIGN

The design of the system is guided by the conceptual structure shown in Fig. 1. The main concepts involved in the design are *animal behavior*, *robotics*, *simulation*, and *programming*. The arrows in the figure show how these concepts interact with each other within the system. In one direction, learners leverage their programming skills to make use of the robotics and simulation components. Through

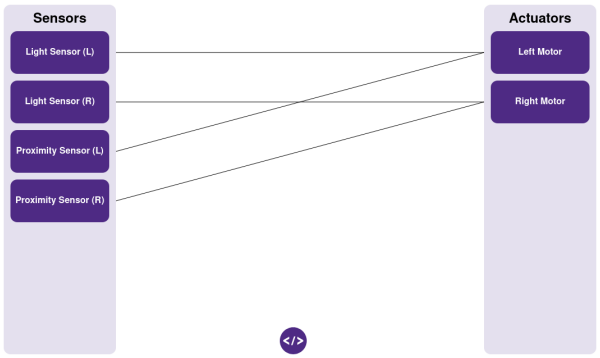


Figure 3: The programming interface.

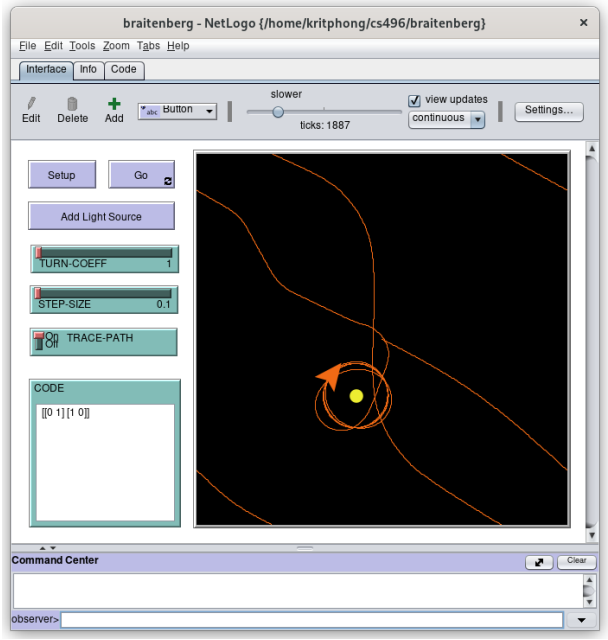


Figure 4: The Braitenberg vehicle simulator.

this process, they gain new understandings of animal behaviors. In the opposite direction, learners use knowledge about animal behavior as a foundation. This knowledge then guides their decision about how to best program the robots and simulator to match the behavior of their animal models. In doing so, they learn about programming strategies that can be used to realize their designs.

4.1 Programming Interface

Velo provides a web-based programming interface that learners can use to program both the robots and the simulation. The interface consists of two columns of *sensors* and *actuators*. A *sensor* represents a sensor that the robot could read data from. Similarly, an *actuator* represents a device that the robot can use to perform an action. In the current design iteration, the *sensors* included are light sensors and proximity sensors. For *actuators*, only motors are included. However, the system can be easily configured to support other kinds of *sensors* and *actuators*.

Programming is done when children connect *sensors* to *actuators*. Each *sensor* can provide its input to multiple *actuators*. Likewise, an *actuator* can receive inputs from multiple *sensors*. Programs are executed by continuously reading values from each *sensor*, and using the read value to determine how *actuators* connected to it should be activated. In the case of motors, the higher the values read from the *sensors*, the faster it turns. If an *actuator* has multiple *sensor* connections, the *sensor* with the highest reading is used.

Making these connections is the only programming construct available. The programming environment does not rely on common programming constructs such as loops, conditionals, or sequencing. This has two main benefits. First, the simplicity makes the programming environment easy to learn. Second, it ensures that all programs built in this environment adhere to the model of Braitenberg Vehicles.

4.2 Simulator

The simulator is implemented in NetLogo[14]. The interface contains buttons that can be used to initialize the simulation, add light sources, and run the simulation. Light sources can be moved around freely through drag-and-drop operations. There are two sliders that control two variables: `TURN-COEFF` and `STEP-SIZE`. `TURN-COEFF` controls how much a motor causes the vehicle to turn with respect to the amount of light measured by its corresponding sensors. `STEP-SIZE` controls how much the vehicle moves at each time step, i.e. how fast it moves. The text box labeled `CODE` is used to configure the connections between light sensors and motors on the vehicle.

4.3 Robot

The robotic platform is designed to be the physical embodiment of the vehicles simulated in the NetLogo environment. The body of the robot consists of a hexagonal base, six vertical walls with holes for mounting sensors and motors, 3D printed motor-mount housings for the DC motors, and an open hexagonal top rack for mounting the GoGo Board 6 [12]. The prototype shown in Fig. 5 is constructed using laser-cut acrylic pieces, but is designed to be modular in nature, allowing users to build it out of any flat material available to them with the help of the design templates. The locations of the holes, save for those necessary to mount the motors on each wall, can be chosen arbitrarily to further the number of possible designs and interactions achievable.

5 CONCLUSIONS AND FUTURE WORK

We presented Velo, a hybrid robotics-simulation learning experience that helps children bridge their interest in animals with programming. Velo consists of three main components: the robot, the simulator, and the programming environment. The programming environment provides a single programming construct, i.e. connections between *sensors* and *actuators*, that ensures all programs constructed this way are compatible with the model of Braitenberg vehicles. The simulator helps learners quickly test and iterate on their ideas, and the robot allows them to realize their visions in the real world. Velo is designed to be used in a curriculum that involves

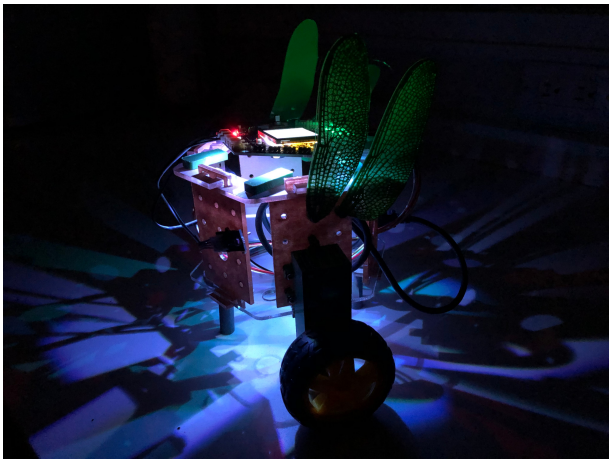


Figure 5: The robotic platform.

analysis and modeling of animal behaviors. By engaging in activities in the curriculum, using Velo as the platform, learners will become aware of the connection between animal behavior, robotics, and programming, as well as develop a number of practical skills in these areas.

In future iterations, we plan to make it possible to code more complex behaviors by implementing functionality to represent non-linear relationships between *sensors* and *actuators*. This would increase the flexibility of programs that could be created, making it possible to model more complex animal behaviors [1]. We are also looking to extend the simulator to support multiple vehicles, allowing learners to explore interactions between vehicles.

ACKNOWLEDGMENTS

We would like to thank Uri Wilensky, Jacob Kelter, Yinmiao Li, and Lexie Zhao for their help and support along the way.

REFERENCES

[1] Valentino Braitenberg. 1986. *Vehicles: Experiments in synthetic psychology*. MIT press.
[2] Po-Yao Chao. 2016. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education* 95 (2016), 202–215. https://doi.org/10.1016/j.compedu.2016.01.010
[3] Neil Fraser. 2015. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000
[4] Michael S. Horn, Jeremy Baker, and Uri J. Wilensky. 2020. NetTango Web. https://netlogoweb.org/nettango-builder
[5] Michael S. Horn, Corey Brady, Arthur Hjorth, Aditi Wagh, and Uri Wilensky. 2014. Frog Pond: A Codefirst Learning Environment on Evolution and Natural Selection. In *Proceedings of the 2014 Conference on Interaction Design and Children* (Aarhus, Denmark) *(IDC '14)*. Association for Computing Machinery, New York, NY, USA, 357–360. https://doi.org/10.1145/2593968.2610491
[6] Stephen R. Kellert. 1985. Attitudes toward Animals: Age-Related Development among Children. *The Journal of Environmental Education* 16, 3 (1985), 29–39. https://doi.org/10.1080/00958964.1985.9942709 arXiv:https://doi.org/10.1080/00958964.1985.9942709
[7] D Mioduser, ST Levy, and V Talis. 2002. Kindergarten children's perception of robotic-control rules. In *Intl Conf Learning Sciences*.
[8] Izabel C Olson and Michael S Horn. 2011. Modeling on the table: agent-based modeling in elementary school with NetTango. In *Proceedings of the 10th International Conference on Interaction Design and Children*. 189–192.
[9] Seymour Papert and Idit Harel. 1990. Situating constructionism. (1990).
[10] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.
[11] Eric Schweikardt and Mark D Gross. 2008. Learning about complexity with modular robots. In *2008 Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*. IEEE, 116–123.
[12] Arnan Sipitakiat, Paulo Blikstein, and David P Cavallo. 2004. GoGo board: augmenting programmable bricks for economically challenged audiences. (2004).
[13] José-Manuel Sáez-López, Marcos Román-González, and Esteban Vázquez-Cano. 2016. Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education* 97 (2016), 129–141. https://doi.org/10.1016/j.compedu.2016.03.003
[14] Seth Tisue and Uri Wilensky. 2004. NetLogo: Design and implementation of a multi-agent modeling environment. In *Proceedings of the Agent*, Vol. 2004. Springer Cham, Switzerland, 7–9.
[15] Sokratis Tselegkaridis and Theodosios Sapounidis. 2021. Simulators in educational robotics: A review. *Education Sciences* 11, 1 (2021), 11.
[16] Uri Wilensky and Seymour Papert. 2010. Restructurations: Reformulations of knowledge disciplines through new representational forms. *Constructionism* 17 (2010), 1–15.
[17] Michelle Hoda. Wilkerson-Jerde. 2012. The DeltaTick Project: Learning Quantitative Change in Complex Systems with Expressive Technologies.

A DEMO PRESENTATION FORMAT

A video detailing how the demo will be presented can be found here: https://drive.google.com/file/d/1-dagh80vk1uDtREz9Es1Y_rO2QlYJWlK/view?usp=sharing.