We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

4,100
Open access books available

116,000
International authors and editors

120M
Downloads

154
Countries delivered to

Our authors are among the
TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Multiplayer Disciplinarily-Integrated Agent-Based Games: SURGE Gameblox

Douglas B. Clark, Paul Medlock-Walton,
Raúl Boquín and Eric Klopfer

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.73051

Abstract

Early work on disciplinary-integrated games (DIGs) focused on Cartesian time-series analyses as the formal representations through which the game communicates challenges and opportunities to the players as well as the formal representations through which the players control the game. In our earlier work, we explored the potential generalizability of the DIG genre in terms of hypothetical examples in physics, biology, chemistry, and the social sciences as well as in terms of multiple model types including constraint-system analyses, system dynamics models, situation-action models, and agent-based models. In particular, Sengupta and Clark and Krinks, Sengupta, and Clark explored the integration of computational modeling, physical models, and Cartesian models. Building on that work, we began outlining theoretical frameworks and arguments highlighting the affordances of moving DIG design more deeply into agent-based modeling. In the current paper, we present the actual design process and rationale through which we developed prototypes of two multiplayer DIG prototypes with agent-based models as the mode of control wherein players create, trade, and elaborate on one another's code as part of gameplay. We close with a discussion of implications for the design of disciplinary-integrated games leveraging agent-based modeling as the focal formal representation for communication and control.

Keywords: science education, science as practice, modeling, agent-based modeling, disciplinary-integrated games, digital games

1. Introduction

Over the past decade, digital games have been the object of substantial research and interest for approaching the difficult task of reorganizing students' conceptual resources [1–3]. Developing environments from Science as Practice perspectives [4–7] that engage students in modeling

INTECH
open science | open minds

© 2018 The Author(s). Licensee InTech. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

would seem to hold great promise. Toward these goals, we have proposed Disciplinarily-Integrated Games as one such approach [8–9]. As proposed in Clark et al. [8], disciplinary integration is an approach to designing games in terms of a science as practice perspective. More specifically, disciplinary integration can be thought of in terms of "model types" and "modeling strategies" [10], which Collins and colleagues have termed "epistemic forms" and "epistemic games" in earlier work [11–13]. Collins and colleagues argue that the professional work of scientists can be understood in terms of model types (epistemic forms) that are the target structures guiding scientific inquiry and modeling strategies (epistemic games) that are the sets of rules and strategies for creating, manipulating, and refining those model types.

DIGs are designed to engage students more deeply in specific modeling and representational practices of developing, interpreting, manipulating, and translating across specific model types. Essentially, all DIGs have the following characteristics: (a) formal representations for controlling the game, (b) formal representations for communicating challenges and opportunities, (c) a phenomenological representation presenting the phenomenon being modeled, (d) an intermediate aggregating representation, and (e) game mechanics and goals focused on engaging the player in interpreting, creating, modifying, and translating across these formal and phenomenological representations.

Early work on DIGs focused on Cartesian time-series analyses as the formal representations for communicating challenges and opportunities as well as the formal representations through which players control the game [8–9]. To have value, the proposed DIG genre must be generalizable. In our earlier work, we explored this claim of generalizability in terms of hypothetical examples of DIGs in physics, biology, chemistry, and the social sciences as well as in terms of multiple model types including time-series analyses with Cartesian formal representations, constraint-system analyses with Cartesian formal representations, and other model types and non-Cartesian formal representations such as system dynamics models, situation-action models, and agent-based models [14]. In particular, Sengupta and Clark [15] and Krinks et al. [16] explored the integration of computational modeling, physical models, and Cartesian models. Clark et al. [17–19] outlined theoretical frameworks and arguments highlighting the affordances of moving DIG design more deeply into agent-based modeling as the formal representational form through which the game communicates challenges and goals to players as well as the formal representational system through which players control the game. In the current paper, we present the actual design process through which we developed prototypes aligning with those ideas for SURGE Gameblox. More specifically, we introduce and describe the design process through which we developed two multiplayer DIG prototypes with agent-based models as the mode of communication and control wherein players create, trade, and elaborate on one another's code as part of gameplay.

2. Technical design

2.1. Gameblox and Starlogo integration to create hybrid SURGE Gameblox

The current project, which we call SURGE Gameblox, is built upon the integration of two pre-existing robust programming environments for students. This process of integration to create the hybrid SURGE Gameblox environment was funded by the SURGE grant from the US National Science Foundation. Essentially, when the project began, there were two preexisting platforms that each had pieces of functionality that were required for the creating the models that we wanted students to modify. These two environments, Gameblox and Starlogo Nova, had been created under prior funding by the MIT Scheller Teacher Education Program. Gameblox is a blocks-based programming language that makes it easy to create 2D multiplayer games. Gameblox is targeted toward making games with a moderate number of sprites, often using a physics engine for games like platformers and mobile experiences including scavenger hunts and participatory models. Gameblox was not built to handle large models with hundreds of agents running simultaneously. Gameblox, however, already has a robust infrastructure for creating multiplayer games. There are blocks that allow for variables to be shared across multiple computers and updated in real time, procedures that can be called on a remote machine, and the ability to create separate instances of a game, allowing independent games to occur between multiple computers.

Starlogo Nova allows students to create 3D agent-based models by programming breeds using a blocks-based language. These agent-based models allow hundreds or thousands of agents to move simultaneously, with each following its own sets of programming instructions. While Starlogo Nova provides a system for handling the complex systems well, it did not have a way to easily connect two models together over a network.

After evaluating the complexity of integrating the two platforms, it was decided to take the 3D renderer from Starlogo and integrate it into the multiplayer framework of Gameblox to allow models to work across multiple computers.

2.2. Collaboration

When the project began, neither Gameblox nor Starlogo had support for allowing more than one user to edit a project at a time. Through a separate funding source, Gameblox was already working on multi-user collaboration in the editor, where two people can edit a project simultaneously, like Google docs. For the SURGE Gameblox project, we set up a separate server that contains this code, which allows users to work on only one block page. Gameblox projects are organized into block pages, similar to how large coding projects are separated into multiple files. The collaboration code is set up so that all editors can view the code on all of the block pages, but each editor can modify only one block page. The projects are set up in such a way that Player 1 can edit only Player 1's code, but can view Player 2's code as it is being changed. This functionality is central to how both the complex systems and physics model activities work.

3. Ecology complex system prototype design

When building this model, we wanted to build on the complex system models that have been developed over the years with Starlogo and to have the model include elements where students are reading data from graphs, using code to modify the model, and having the model meaningfully work across multiple devices.

3.1. Existing Starlogo models

We started by looking at Starlogo models that are used now for complex systems, with two exemplars being the colored butterflies and the fish and algae model. In both of these models, students are able to modify how the models run by changing the values on sliders that affect particular breeds and their environment. In the butterflies model, the sliders are used to set the initial conditions for the model. For the SURGE Gameblox project, however, the variability achieved in the models needs to occur through manipulation of code when the model is designed instead of through the on-screen widgets while the model is being run. In addition, given that many of the graphs students are reading occur over time, unlike the existing Starlogo models, it is important that values change throughout the running of the model as opposed to only during the setting of initial conditions.

An important functionality that we carried over from more complex models, like the fish and algae model, is the use of traits, properties that are associated with the instances of classes of objects that are used to store information about the agents as the model runs.

3.2. Prototype #1: Positioning plants to maximize animals on a farm

3.2.1. Design

Players compete to get as many animals living on their farms as possible (**Figure 1**). The animals roam between the two farms, pause to eat grass, and get varying levels of energy from each type of grass. Each player programs how much grass of each type to add to his or her farm by reading graphs that inform the user how much energy each animal gets, and the percentage of type of animals in each level. Players have graphs of the seasonal changes in sunlight and rainfall. Each type of grass does better in different climate conditions. Players need to account for the seasonal changes represented in these graphs to

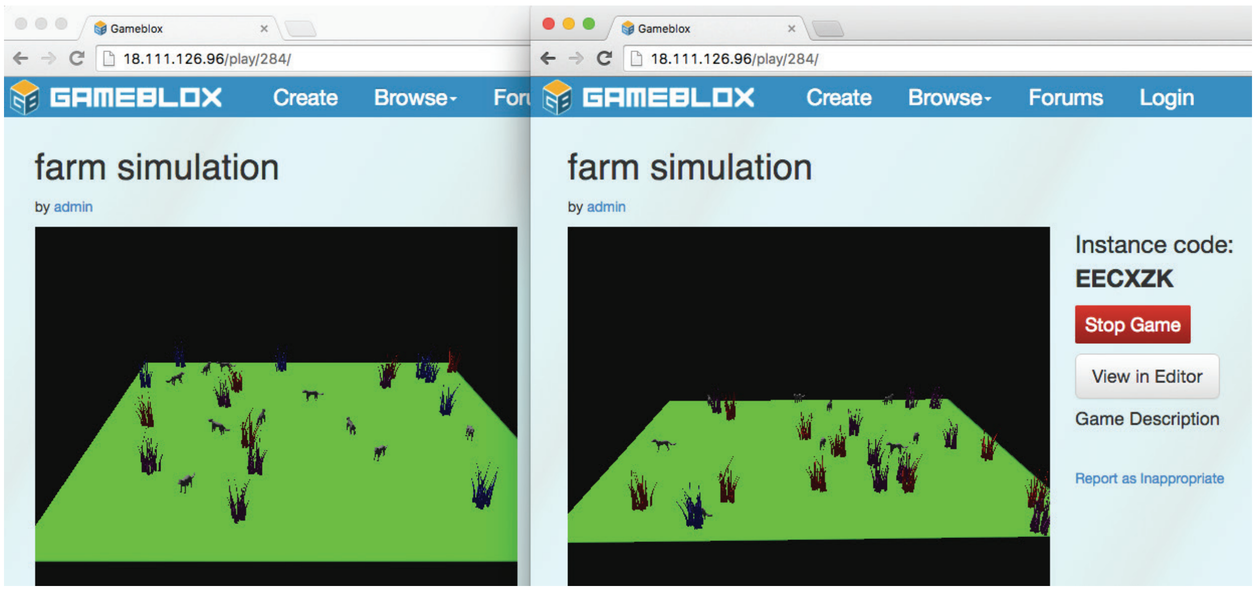


Figure 1. Screenshot of farm game for two players.

maximize their populations. In our initial version of the prototype, we focused on a competitive structure for this prototype so that we could compare competitive and collaborative structures. The player who maintains the largest number of animals on his or her side wins (**Figures 2** and **3**).

3.2.2. Feedback

The two largest pieces of feedback we received were that it was hard to know what location was best and that you could not react dynamically to weather changes.

Strategy behind setting positions not apparent. When playing the game, the animals walk randomly from one screen to another, and the plants are placed at random y locations at the particular x value selected by the user. Given that the animals are randomly placed on the board, the locations where the animals go are almost at random. This led to there being no good strategy for optimizing where a player places plants to grow more effectively. One could imagine a future version having a heat map that showed the climate in different areas of the board and allowing the players to read that map to make strategic decisions around where to plant, but this version was not implemented.

Cannot react dynamically to weather changes. The coding in this prototype was limited to planting different types of grass based on the time that had passed. This not only limited the complexity of the code that could be written, but also meant that the model could run only for the limited period of time on the graphs, and players could not extrapolate their knowledge to handle situations that were not available on a graph.

3.3. Prototype #2: Generalizing rules from the graphs

The two main changes that occurred for the second version of the prototype were to (a) remove the position selection from the plant block, so all plants would generate in random locations and (b) include a block that would allow the player to get data from the graph about a particular point in time, as shown in **Figure 4**.

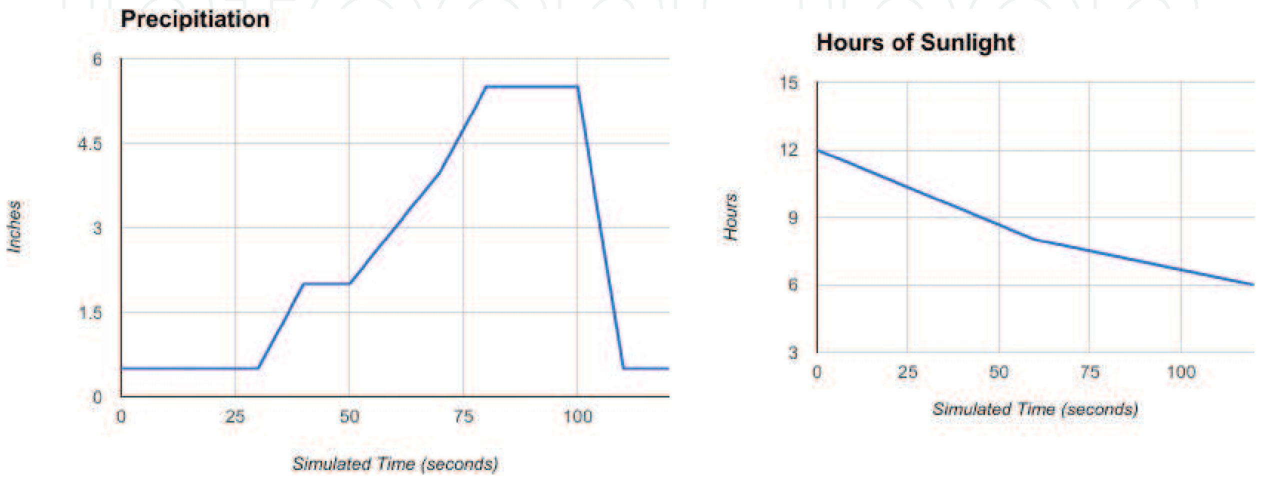
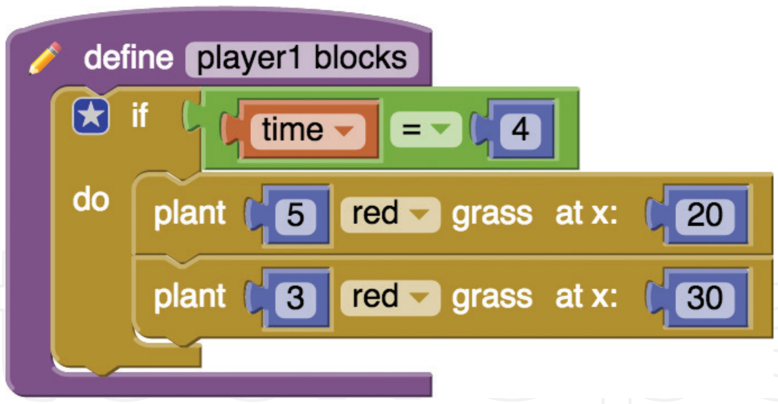
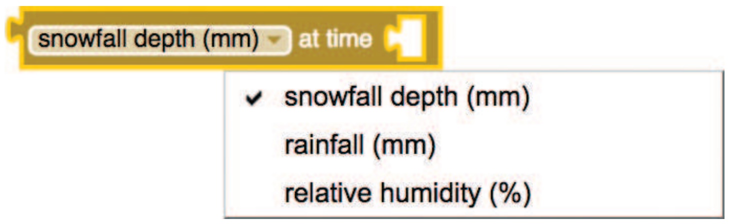


Figure 2. Graphs of precipitation and sunlight.



```
define player1 blocks
  if time = 4
  do plant 5 red grass at x: 20
     plant 3 red grass at x: 30
```

Figure 3. Screenshot of simple blocks to place grass on a location.



```
snowfall depth (mm) at time
```

Figure 4. Block used to get data from graph.

The addition of these changes allows the activity to be broken up into two parts. The first part of the activity is run similar to the first prototype, where different colored grass is placed randomly on the ground by choosing specific times to plant. The second part of the activity, however, allows for more complicated coding strategies, enabling the user to decide what to plant based on the values of the rainfall and sunlight instead of simply choosing static times.

3.4. Prototype #2: Next steps

The next iteration of the prototype will focus on revising features of the game activity structure to focus on a collaborative game rather than a competitive game. Our initial goals involved creating one competitive prototype (ecosystems) and one collaborative prototype (kinematics) to be able to explore the affordances and trade-offs between competition and collaboration. We therefore created the competition version of ecosystems first, but our ultimate commitments are toward building collaborative environments. Recent meta-analyses of digital games and learning suggest that collaborative structures tend to be more effective for learning [20–21], and collaborative structures align very well with perspectives framing students as modelers [4–5] and computational thinkers [22–23]. As Kafai et al. [24] articulate, CT can and should be framed in terms collaborative design and production within communities. Therefore, while we believe that comparison analyses between competition and collaboration are of value, we ultimately think that the research and theoretical foundations around games for learning as well as computational participation and science as practice provide greater support for collaborative approaches.

IntechOpen

4. Newtonian kinematics prototype design

We chose Newtonian kinematics as the domain for the second prototype, and we chose a collaborative structure. We made these choices so that we could compare differences in collaborative versus collaborative structures as well as compare across domains. When creating the physics model, we had three goals in mind: 1) students should use code to collaboratively modify the model based on reading graphs, 2) the model spans the screens on two computers, and 3) the graphs should describe a phenomenon in physics that students would study in class.

4.1. Prototype #1: Coloring cubes

Our first prototype had cubes moving based on speed vs. time graphs given to the player. The player needs to predict the location of the cube at a particular point in time.

4.1.1. Gameplay

Players work collaboratively to turn all of the moving cubes purple. Each of the cubes moves based on position or velocity graphs given to the students. Students then need to predict when the cubes will cross the center of screen. When the cube reaches the right side of one player's screen, it will appear on the left-hand side of the other player's screen. Player 1 uses the blocks to fire the red laser, which turns cubes red, and player 2 controls the blue laser. When a cube is already red and is hit by the blue laser, it turns purple. Players win by turning all of the cubes purple in the most efficient way (**Figures 5–7**).

4.1.2. Limitations

The two largest problems that we noticed through testing this prototype were that (a) the prototype did not incentivize collaboration sufficiently, depending on the pairs working on it, and (b) the absence of numbers on the 3D renderer caused confusion.

Not collaborative. The largest problem with this prototype was that each player could independently solve for the location of the cube and did not need to talk to the other player. This defeated one of the large goals of the project, and we knew therefore that we needed to push the prototype in a different direction.

Tick marks. The 3D renderer that we are using from Starlogo is limited in that we can put only 3D objects in the model. Although we were able to verbally inform the players about the axis on the green floor, it was hard for a player to plan and adjust their strategies and code without markings and numbers to know exactly where the cubes are at a particular time. We therefore chose to move to a 2D environment where we could label the positions in our next iteration.

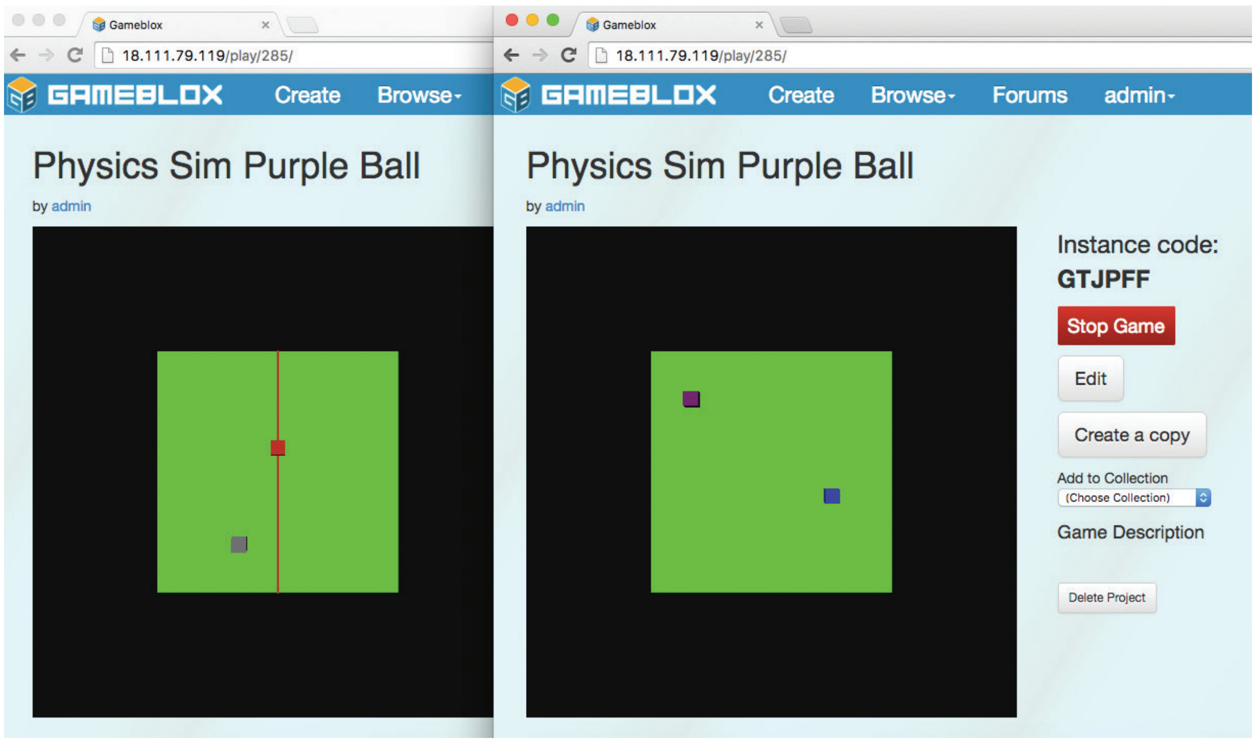


Figure 5. Running simulation.

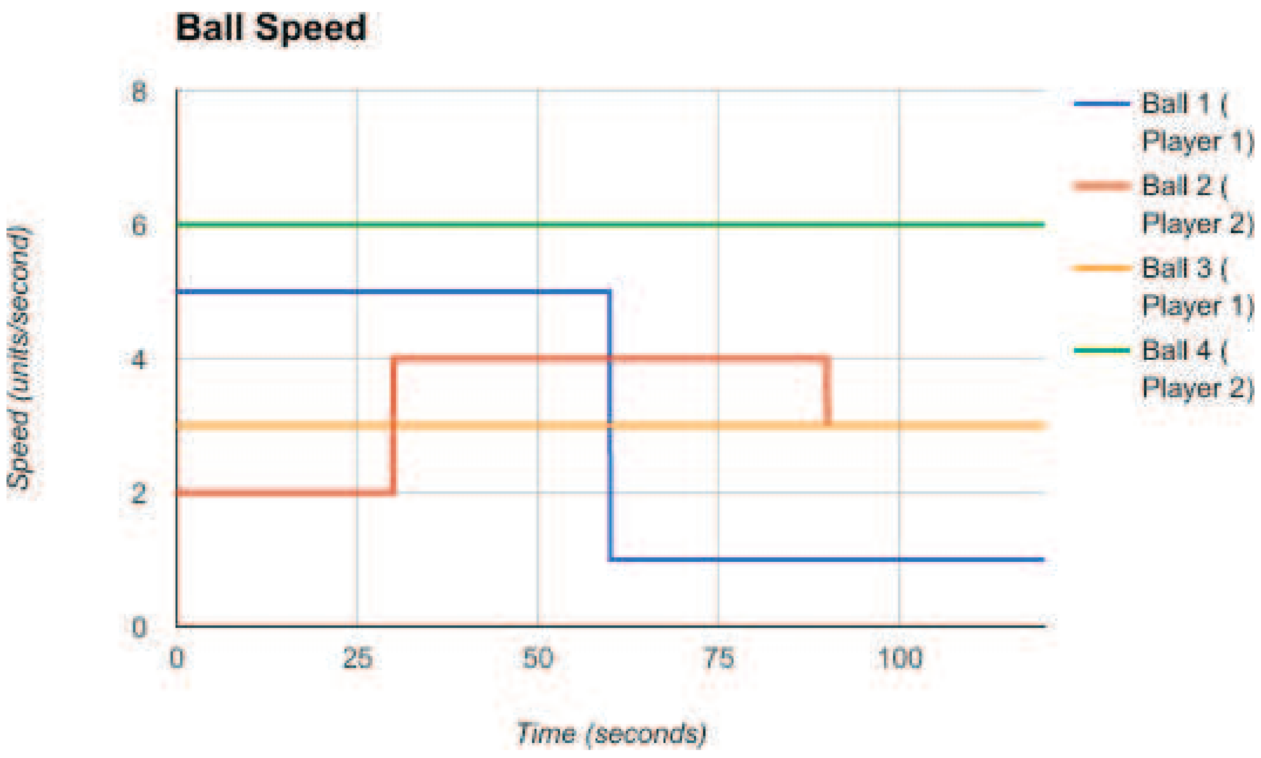


Figure 6. Graphs of the speed of the balls over time.

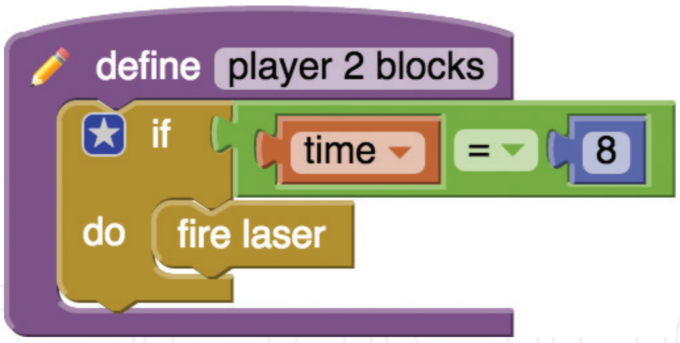


Figure 7. The simplest block needed for a player to color a single cube at a particular point in time.

4.2. Prototype #2: Moving across a marked line

Given that the underlying gameplay was not collaborative, we shifted to creating paper prototypes that would lead to better dynamics within the model. After several rounds of designing and playtesting various prototypes, the one described below became the starting point for the next digital version of the project.

4.2.1. Gameplay

Each player has a timeline and a graph. At the beginning of the game, the player learns where the ball should land at a particular time. In the figures, for example, the ball should be at $x = 10$ when $t = 10$. Player 1's speed vs. time graph applies only when the ball starts moving on the line controlled by player 1 (when x is between 1 and 5). The same is true for player 2 (when x is between 6 and 10). Players can choose to add a "Speed Up" or a "Slow Down" when the ball starts moving on their part of the line. The "Speed Up" or "Slow Down" will increase or decrease the speed of the ball by 2 (**Figures 8** and **9**).

4.2.2. Feedback

When testing the game, the three pieces of feedback we consistently saw were that (1) collaboration naturally occurred as both players were trying to solve the problem, (2) the game was harder than expected, and (3) there was confusion on how to read the graphs (**Figure 10**).

Collaboration. One problem with the previous prototype was that each player could individually solve for the location of the cube without needing to speak with the other player in order to succeed. In this prototype, that is no longer the case, because by adding a "speed up" the ball will appear in a different location at a different time on the other player's screen, which will cause him or her to change strategy.

Harder than expected. The same benefit that made collaboration happen, where a small change would affect the strategy of the other player, made the game significantly harder than we

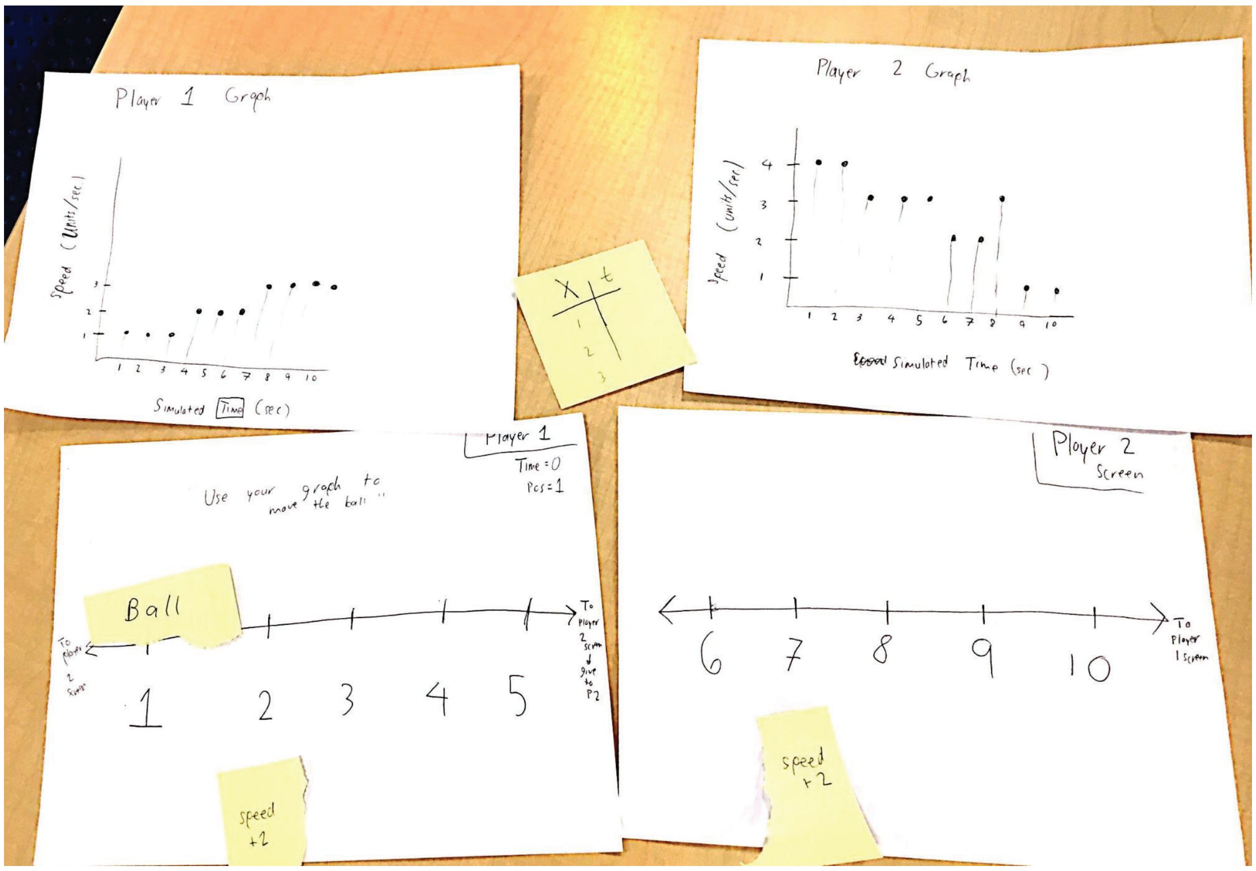


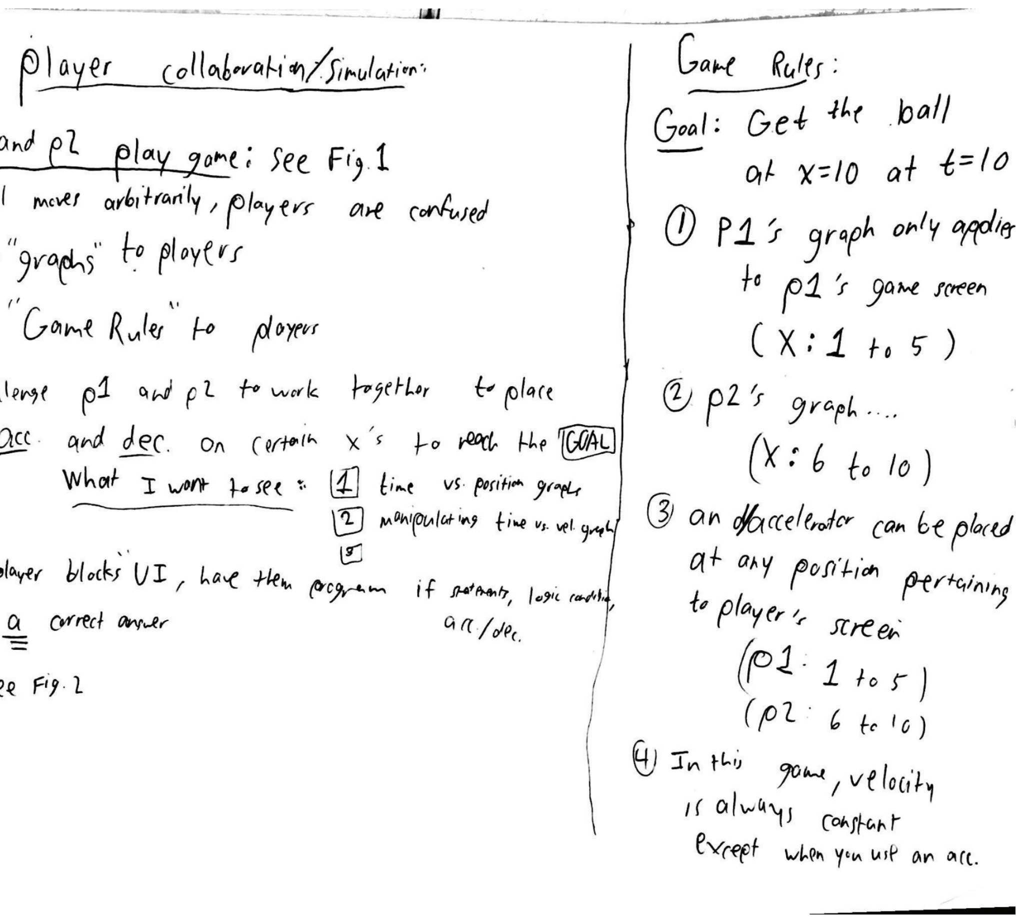
Figure 8. Game setup in paper prototype.

expected. This had both unexpected benefits and drawbacks: students could spend a very long time (20+ minutes) trying to find a solution, and if they did not solve it systematically, it could lead to frustration. The most successful players wrote a table of times and locations of the ball over time. Having the players generate a table that leads to a position graph achieved an outcome we did not expect.

Confusion reading the graph. Some students were confused by the points on the graph representing the speed at a particular point as shown in **Figure 8**. Other students were confused by graphs in **Figure 11** because they were trying to find the velocity at a point with a vertical line (the instantaneous change) but were not sure which value they should choose. Ultimately, we used the graphs in **Figure 11** because they are both more accurate and are more similar to the formal Cartesian representations that students encounter in other science contexts.

4.2.3. Moving to a digital prototype

While we were originally expecting to use the paper prototypes to simply test the interactions before moving to a digital version, testing demonstrated that working and coordinating on paper first was important to the learning experience, especially given the difficulty of the challenge. After



Player collaboration/Simulation:

1. p1 and p2 play game: see Fig 1
   Ball moves arbitrarily, players are confused
2. Show "graphs" to players
3. Show "Game Rules" to players
4. Challenge p1 and p2 to work togethor to place acc. and dec. on certain x's to reach the $\boxed{\text{GOAL}}$
   What I want to see: $\boxed{1}$ time vs. position graphs
   $\boxed{2}$ manipulating time vs. vel. graph
   $\boxed{3}$
5. Show "player blocks" UI, have them program if statements, logic condition, acc/dec.
6. Find a correct answer

   See Fig. 2

Game Rules:

Goal: Get the ball at x=10 at t=10

1. P1's graph only applies to p1's game screen
   (X: 1 to 5)
2. p2's graph....
   (X: 6 to 10)
3. an ~~d~~accelerator can be placed at any position pertaining to player's screen
   (p1: 1 to 5)
   (p2: 6 to 10)
4. In this game, velocity is always constant except when you use an acc.

Figure 9. Instructions from paper prototype.

the students believed they had found a solution of "speeds ups" and "slow downs," we would provide them a sheet where each player would write his or her solution, like the one in **Figure 12**.

The students would then see a Gameblox model like the one below, which simulates the same scenario that they saw on paper (**Figure 13**).

Each player would then be presented with a limited set of blocks to program the "speed ups" and "slow downs" that they discovered from solving the paper version. If the students got the answer correct, then the dinosaur would arrive at the desired location after 10 seconds had passed. If not, then the students would know that their solutions were incorrect (**Figure 14**).

4.2.4. Expanding the game

After we had the basic mechanic of getting the ball to go to a particular location at a particular time, there were several changes we made to expand the work the players had already done, including graphing how the ball moves, and calculating the distance it covered.

Graphing actual speed vs. time. As the ball moves from one half of the line to the other (between 0 and 5 and 6–10), the speed is dependent on the graph of the particular player who is

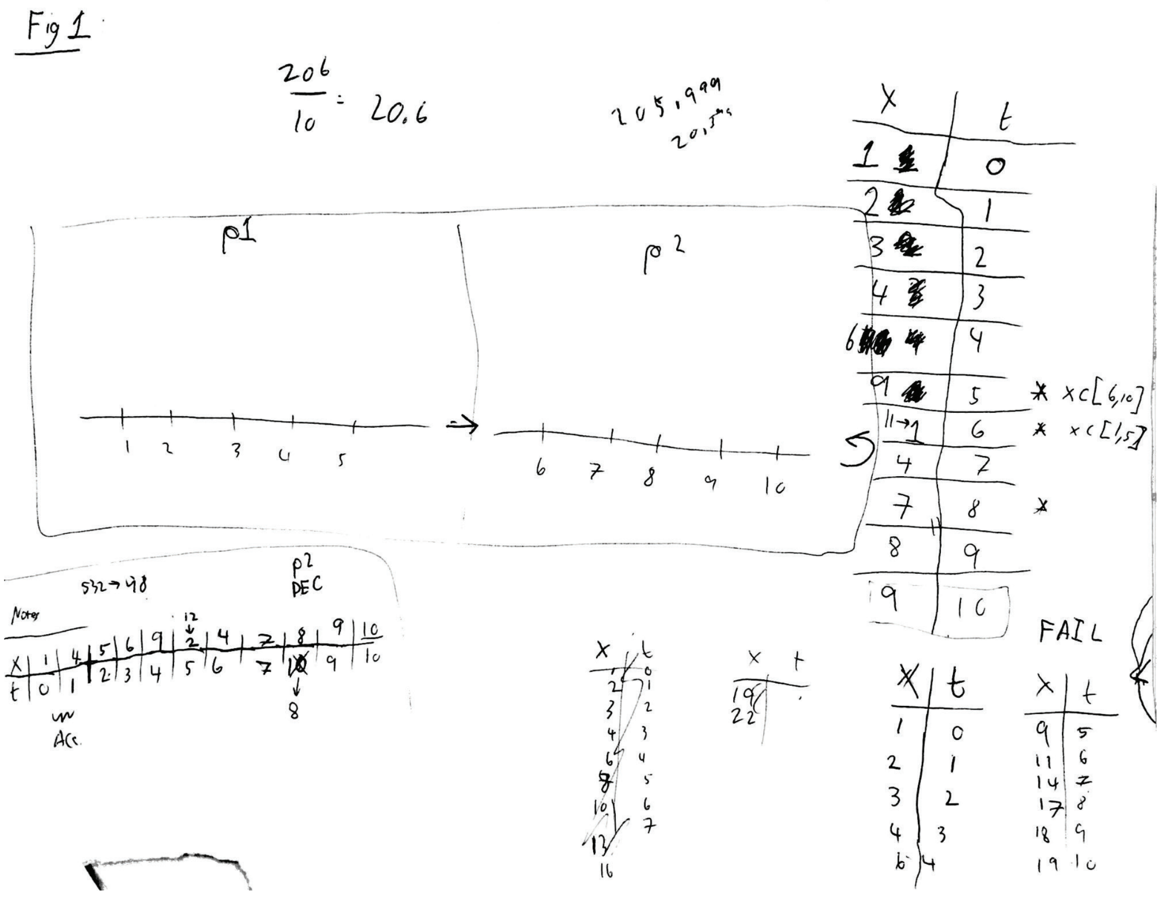


Figure 10. Table of time and position values.

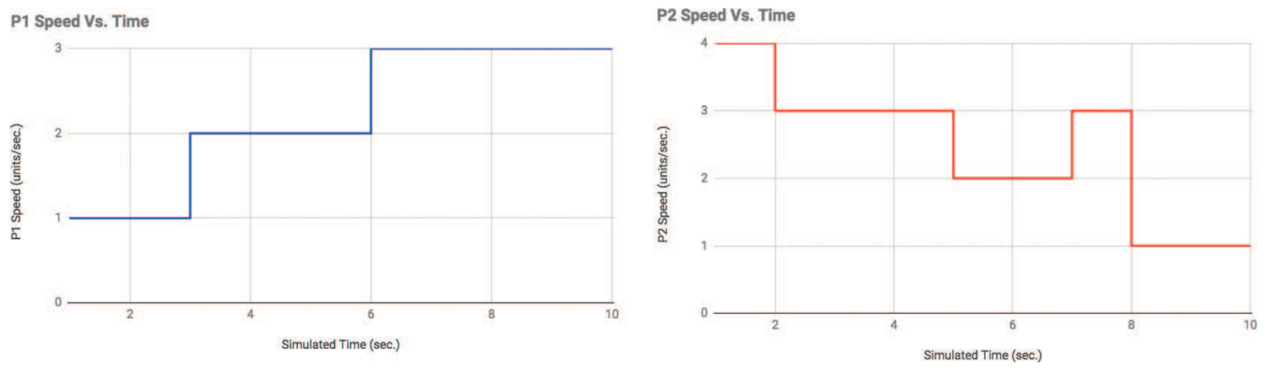


Figure 11. Speed vs. time graphs for player 1 & 2.

programming for that part of the line. After the students have programmed their solutions in Gameblox, they create the graph of the speed vs. time of the ball. This means following where the ball is and knowing whether to reference Player 1's or 2's graph in order to find the ball's speed at the appropriate time.

Calculating distance. With a graph of the actual speed of the ball over time, it is possible to calculate the total distance the ball has traveled by calculating the area underneath the

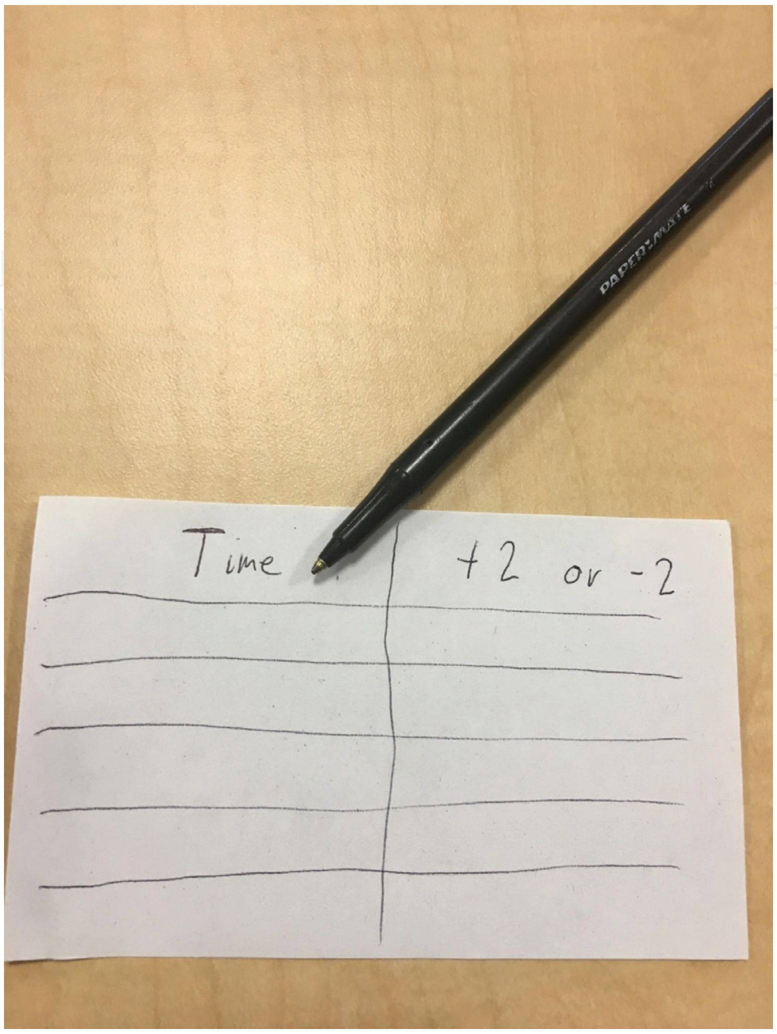


Figure 12. Time and “speed up”/“slow down” table.

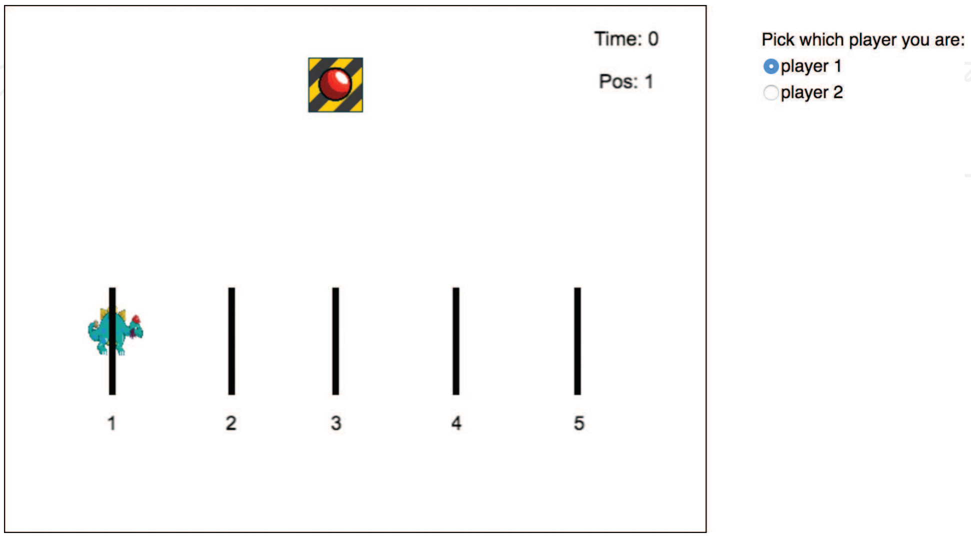


Figure 13. Gameblox version of physics simulation.

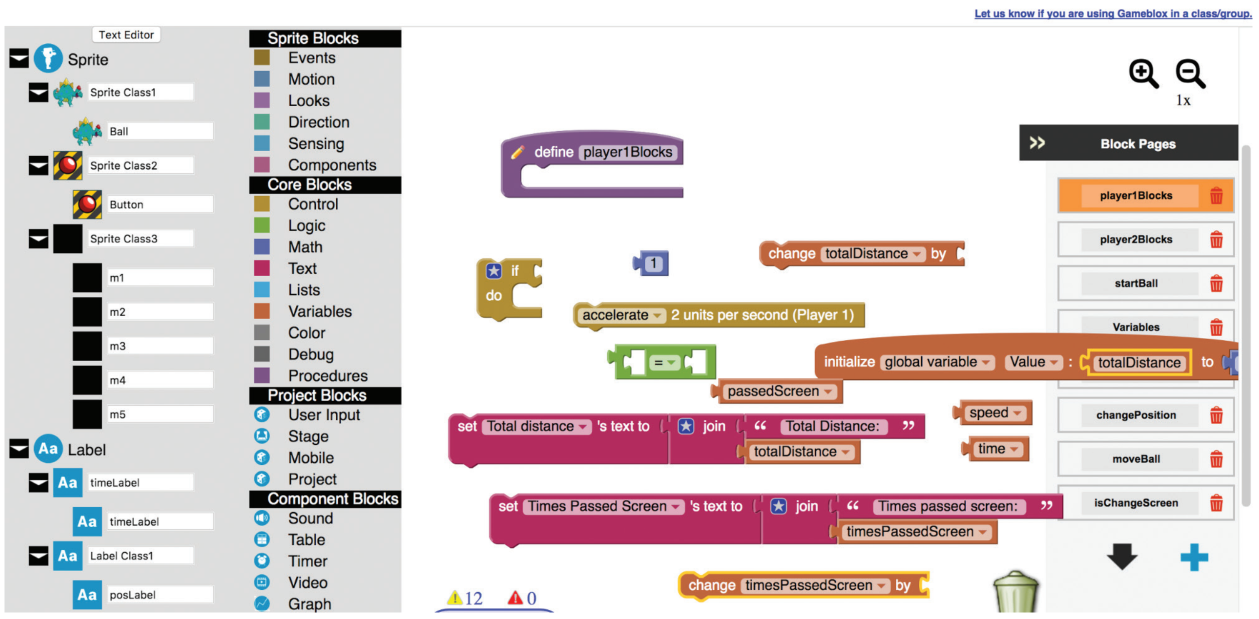


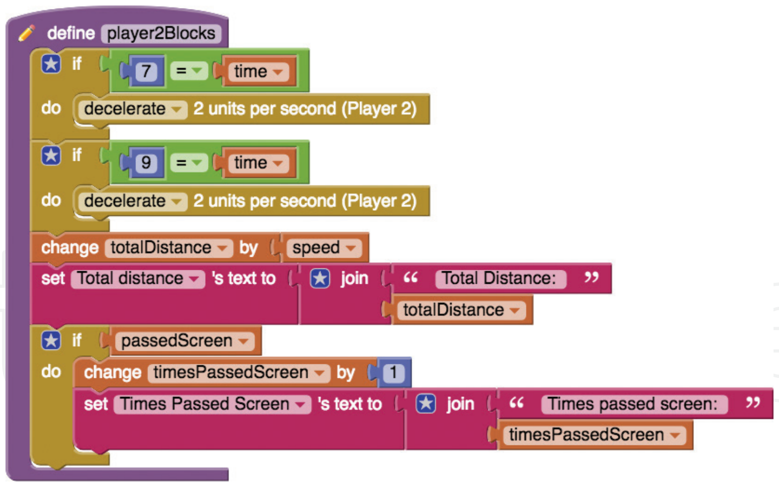
Figure 14. All of the blocks that the users are able to choose from to construct their solutions.

graph. In early versions of the prototype, we had speeds that could be 1 or 0. This meant the player could use a "slow down" and the speeds would be −1 or −2. The model would correctly have the character move backward on the line, demonstrating a negative speed. There was confusion, however, regarding whether the negative area on the graph should be added or subtracted from the total distance sum when calculating the distance. This can help lead to discussions around the difference between distance and displacement, as the decision of how one handles the area with a negative speed changes the quantity that the players obtain.

Programming the distance. After the students determined the distance that the ball traveled through calculating the area underneath the combined graph, we had the players program a short script in Gameblox to programmatically determine the distance covered. Since d = vt, and t = 1 for every step, the equation simplifies to d = v, so each player needs to add the speed that the ball is traveling every time his or her script is triggered. By doing this, each player can calculate how far the ball traveled from their screen. By adding the distances from both players, they are able to see the total distance the ball has traveled. If the players are then asked about how to calculate the displacement, they would need to take the absolute value of the speed. Below is an example of a solution that has two "slow downs" and calculates the distance traveled (**Figure 15**).

4.2.5. Simplifying the experience

Finding the solution to the original problem we proposed took too long for all of our original testing teams and was discouraging to some teams. In order to create an easier on-ramp to the activity, we created a "Level 0," which helps the students understand the mechanics of the problem though an easier set of graphs as shown in **Figure 16**.



```
define player2Blocks
  if 7 = time
  do  decelerate 2 units per second (Player 2)
  if 9 = time
  do  decelerate 2 units per second (Player 2)
  change totalDistance by speed
  set Total distance 's text to join " Total Distance: "
                                     totalDistance
  if passedScreen
  do  change timesPassedScreen by 1
      set Times Passed Screen 's text to join " Times passed screen: "
                                              timesPassedScreen
```

Figure 15. A sample solution of the blocks.

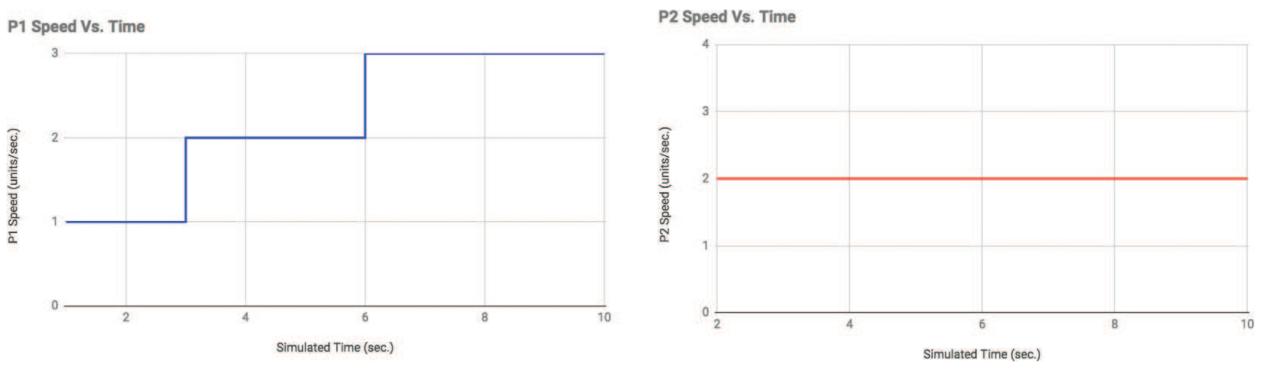


Figure 16. Level 0 graphs for simpler solutions.

The graphs are notable not only for having constant speed but also for having speeds that are two and above. Speeds that are two and greater mean that if a player uses a "Slow down," then the speed will not be negative, which will eliminate possible confusion when calculating distance. Students would go through the entire process of the game with these simpler graphs (finding a solution, programming the solution, creating a combined speed graph, calculating the distance, and programming the distance), before doing the same activity with the more complex graphs.

4.2.6. Next steps for the kinematics prototype

Our sense is that the current structure of this prototype potentially focuses more heavily on a specific solution than we might like. Another related perspective on designing games for science education is the constructible authentic representations (CARs) perspective [25]. CARs is highly related to the DIG perspective in that CARs focuses on (a) meaningful construction, (b) conceptually integrated primitives, and (c) authentic representations. One aspect that the CAR perspective emphasizes very heavily is the "sandbox" aspects of the game, in the sense of incentivizing more open-ended experimentation, than sometimes results in DIGs, where the puzzles at the heart of a level or game can be more focused. We would like for future versions of this prototype to be more "CARs-like" and we will explore structures that incentivize open-ended exploration to a greater degree than in the current prototype, and structures that may feel more game-like than the current prototype.

5. Implications and conclusions

To have value, the proposed DIG genre must be generalizable. As described in the introduction, our earlier work has discussed this claim of generalizability in terms of hypothetical examples of DIGs in physics, biology, chemistry, and the social sciences as well as in terms of multiple model types including time-series analyses with Cartesian formal representations, constraint-system analyses with Cartesian formal representations, and other model types and non-Cartesian formal representations such as system dynamics models, situation-action models, and agent-based models [14]. Sengupta and Clark [15] and Krinks et al. [16] conducted our foundational work on integrating agent-based modeling into DIGs, and Clark et al. [17–19] have outlined theoretical frameworks and arguments highlighting the affordances of moving DIG design more deeply into agent-based modeling. In this current paper, we have presented the actual design process through which we developed prototypes aligning with those ideas for SURGE Gameblox. More specifically, we introduced and described the design process through which we developed two multiplayer DIG prototypes with agent-based modeling as the formal representational form through which the game communicates challenges and goals to players as well as the formal representational system through which players control the game.

We would argue that there are five main implications to draw from our work developing these two prototypes and the SURGE Gameblox environment. Each of these five implications focuses on different aspects and kinds of generalizability.

First, at the most basic level, these prototypes serve as proofs of concept that such games and environments are indeed technically possible. We certainly would not argue that these prototypes represent the only way that such games or environments might be structured, but these prototypes show the potential viability and possibilities of such games and environments.

Second, these prototypes demonstrate the generalizability of the multiplayer agent-based modeling DIG approach pedagogically. Ecosystems and kinematics as domains hail from different disciplines and represent very different kinds of agent-based models in the sense that the ecosystems models focus on complex system dynamics whereas the kinematics model focuses more on discrete systems of single agents. These two prototypes therefore suggest that pedagogically this approach can be generalized across multiple domains and types of systems.

Third, these prototypes demonstrate the generalizability of such an approach economically and pragmatically in the sense that both of these prototypes are built on top of the same core SURGE Gameblox environment. It is certainly true that SURGE Gameblox is built upon the robust infrastructure of MIT's Gameblox and Starlogo Nova software environments, but the resulting hybrid SURGE Gameblox environment is indeed capable of supporting cost-effective and time-effective development of similar DIGs across multiple domains and topics.

Fourth, while the development of DIGs for new topics in disciplines can be cost-effective and time-effective, such development will still always require iterative refinement and testing through the design and development phases. Neither of the two prototypes was completed in the first round of development. Both prototypes required multiple rounds of development from their initial conceptualization to their current structure. Furthermore, the designs will benefit from multiple additional iterative rounds of design, testing, and refinement. This is more than just refining the software code. This iterative development and testing and refinement are also requisite parts of the design process for the activity structures themselves. As discussed in our recent meta-analysis of digital games for learning [20], this implication highlights the fact that design, rather than medium alone, determines the efficacy of a learning environment, whether it be a classroom lab, a textbook, a lecture, or a recreational digital game or a disciplinary-integrated game.

Fifth, and finally, these prototypes provide proof of concept for the broader DIG genre in terms of generalizability. While early work on DIGs focused on Newtonian kinematics with Cartesian timeseries analyses as the formal representations of control and communication, we have argued in our earlier work for the generalizability to other topics and formal representational systems [8, 14, 15]. These prototypes provide proofs of concept of such generalizability of the broader DIG conception of developing disciplinary-integrated games wherein manipulation and translation across formal representational systems provide the means of communication and control within the game as players engage in modeling and computational thinking from a science as practice perspective.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1119290. Earlier development of Starlogo Nova was supported by the National Science Foundation under Grant No. 1508911 and 1639069. 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.

Author details

Douglas B. Clark[1]*, Paul Medlock-Walton[2], Raúl Boquín[2] and Eric Klopfer[2]

*Address all correspondence to: douglas.clark@ucalgary.ca

1 University of Calgary, Calgary, AB, Canada

2 MIT, Cambridge, MA, USA

References

[1] Honey MA, Hilton M, editors. Learning Science through Computer Games and Simulations. National Research Council. Washington, DC: National Academy Press; 2010

[2] Martinez-Garza M, Clark DB, Nelson B. Digital games and the US National Research Council's science proficiency goals. Studies in Science Education. 2013;**49**:170-208. DOI: 10.1080/03057267.2013.839372

[3] Young MF, Slota S, Cutter AB, Jalette G, Mullin G, Lai B, ... Yukhymenko M. Our princess is in another castle: A review of trends in serious gaming for education. Review of Educational Research. 2012;**82**:61-89. DOI: 10.3102/003465431243698

[4] Lehrer R, Schauble L. Cultivating model-based reasoning in science education. In: Sawyer RK, editor. The Cambridge Handbook of the Learning Sciences. Cambridge, England: Cambridge University Press; 2006. pp. 371-388

[5] Lehrer R, Schauble L. Scientific thinking and science literacy: Supporting development in learning in contexts. In: Damon W, Lerner RM, Renninger KA, Sigel IE, editors. Handbook of Child Psychology. 6th ed. Vol. 4. Hoboken, NJ: John Wiley and Sons; 2006

[6] Pickering A. The Mangle of Practice: Time, Agency, and Science. Chicago: University of Chicago Press; 1995

[7] Duschl RA, Schweingruber HA, Shouse AW, editors. Taking Science to School: Learning and Teaching Science in Grades K-8. National Research Council Board on Science Education, Center for Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press; 2007

[8] Clark DB, Sengupta P, Brady C, Martinez-Garza M, Killingsworth S. Disciplinary integration in digital games for science learning. International STEM Education Journal. 2015;**2**(2):1-21. DOI: 10.1186/s40594-014-0014-4. http://www.stemeducationjournal.com/content/pdf/s40594-014-0014-4.pdf

[9] Clark DB, Virk SS, Sengupta P, Brady C, Martinez-Garza M, Krinks K, Killingsworth S, Kinnebrew J, Biswas G, Barnes J, Minstrell J, Nelson B, Slack K, D'Angelo CM. SURGE's evolution deeper into formal representations: The siren's call of popular game-play mechanics. International Journal of Designs for Learning. 2016;**7**(1):107-146. https://scholarworks.iu.edu/journals/index.php/ijdl/article/view/19359

[10] Collins A. What's Worth Teaching?: Rethinking Curriculum in the Age of Technology. New York: Teachers College Press; 2017

[11] Collins A. A study of expert theory formation: The role of model types and domain frameworks. In: Khine MS, Saleh I, editors. Models and Modeling: Cognitive Tools for Scientific Enquiry. London: Springer; 2011. pp. 23-40

[12] Collins A, Ferguson W. Epistemic forms and epistemic games. Educational Psychologist. 1993;**28**:25-42

[13] Morrison D, Collins A. Epistemic fluency and constructivist learning environments. Educational Technology. 1995;**35**(5):39-45

[14] Clark DB, Sengupta P, Virk SS. Disciplinarily-integrated games: Generalizing across domains and model types. In: Russell D, Laffey J, editors. Handbook of Research on Gaming Trends in P-12 Education. Hershey, PA: IGI Global; 2016. pp. 178-194. DOI: 10.4018/978-1-4666-9629-7

[15] Sengupta P, Clark DB. Playing Modeling games in the science classroom: The case for disciplinary integration. Educational Technology. 2016;**56**(3):16-22

[16] Krinks K, Sengupta P, Clark DB. Benchmark lessons, modeling, and programming: Integrating games with modeling in the curriculum. International Journal of Gaming and Computer-Mediated Simulations. in press

[17] Clark DB, Sengupta P. Reconceptualizing Games for Integrating Computational Thinking and Science as Practice: Collaborative Agent-Based Disciplinarily-Integrated Games; (submitted)

[18] Clark DB, Medlock-Walton P, Sengupta P, Brady C, Klopfer E. Prototypes of Collaborative Agent-based Disciplinarily-Integrated Games for Integrated STEM Education. Presentation at International Society for STEM in Education (ISSE) Symposium 2017; Banff, Canada; 2017

[19] Clark DB, Medlock-Walton P, Brady C, Sengupta P, Klopfer E., Duncan F, Krinks K, Virk S. Multi-Player Disciplinarily Integrated Games As C-Thinking. Poster Presented at the Annual Meeting of the American Educational Research Association; San Antonio, TX; 2017

[20] Clark DB, Tanner-Smith EE, Killingsworth SS. Digital games, design, and learning: A systematic review and meta-analysis. Review of Educational Research. 2016;**86**(1):79-122

[21] Wouters P, van Nimwegen C, van Oostendorp H, van der Spek ED. A meta-analysis of the cognitive and motivational effects of serious games. Journal of Educational Psychology. 2013;**105**:249-265. DOI: 10.1037/a0031311

[22] NRC. Report of a Workshop on the Scope and Nature of Computational Thinking. Washington, DC: National Academy Press; 2010

[23] Grover S, Pea R. Computational thinking in K–12: A review of the state of the field. Educational Researcher. 2013;**42**(1):38-43

[24] Kafai YB, Burke Q, Resnick M. Connected Code: Why Children Need to Learn Programming. Cambridge, MA: MIT Press; 2014

[25] Holbert N, Wilensky U. Constructible authentic representations: Designing video games that enable players to utilize knowledge developed in-game to reason about science. Technology, Knowledge and Learning. 2014;**19**(1-2):53-79. DOI: 10.1007/s10758-014-9214-8

IntechOpen

IntechOpen