Computational Thinking and Simulation in Teaching Science and Mathematics

Hasan Shodiev

Abstract Characteristics of scientific phenomenon are commonly investigated using mathematical tools in science and engineering to develop our conceptual understanding. However, computational thinking and modelling with simulations can result in a more advanced understanding of scientific concepts and offer an effective learning experience for students with various backgrounds. In this work, we show how a simulation tool, Scratch, can be used to unfold the abstract side of science through project based visualizations in fun and engaging ways. It can be an effective approach in attracting young talented students to science and technology by motivating their natural imagination to probe scientific abstraction.

1 Introdction

The educational system is lacking in progress in implementing the computational approach to understanding nature and technology. However, the science community has developed a novel method of solving problems by simulating various phenomena in many science and engineering disciplines. This offers us new answers to scientific questions that are different from theory and experimentation. Computational thinking (CT) in addition to critical thinking is very important when utilizing the computational approach. This can be illustrated in Figure 1. where the diagram shows a CT path parallel to critical thinking as an integral part of obtaining solutions.

Computational thinking emerged as a new paradigm alongside mathematical, physical, musical and other types of thinking after the availability of computers. CT in problem solving was first introduced by Dr. Seymour Papert [5]. In 1971, Dr. Papert, showed the use of computational thinking in performing non computational

Hasan Shodiev
Wilfrid Laurier University, 78 University Ave., Waterloo, ON, N2L 3C5e-mail: hshodiev@wlu.ca

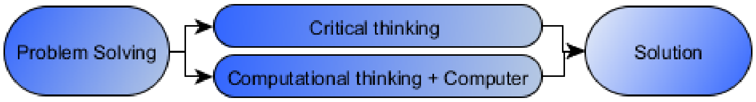
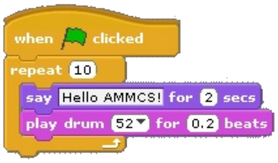


Fig. 1 2-D Problem solving model

activities. In his work, he forged ideas that are at least as explicative as the Euclid-like constructions and turtle geometry but more accessible and more powerful [6].

He defined CT as a problem solving method that uses computer science techniques and concepts. Jeanette Wing [8] recently started reviving CT and emphasizing its role across all disciplines. She argued that computational thinking is a fundamental skill for everyone, not just for computer science majoring students. She initiated a profound engagement with the core questions of what computer science is and what it might contribute to solving problems across the spectrum of human inquiry. We argue that advances in educational technologies allow us to bring computational thinking and effectively use it in secondary and postsecondary school levels. We intend to help bridge the gap between the K-12, non-computer science disciplines and the computer science education communities by investigation of relevant age appropriate resources for science, music [7], art [2] and video games [3]. In this work we propose embedding computational thinking concepts with a universal tool called Scratch [1]. A key component to employing CT with Scratch is the possibility to visualize the phenomena which allows enhanced understanding of the concept. There is no clear evidence of using Scratch by young women. The US National Center for Women and Information Technology (NCWIT) in a case study about Scratch, calls Scratch a promising practice for increasing gender diversity in information technology [4]. Programs in Scratch can be created by simply snapping together graphical blocks, much like LEGO bricks or puzzle pieces (see Figure 2).



```
when 🏁 clicked
repeat 10
   say Hello AMMCS! for 2 secs
   play drum 52▾ for 0.2 beats
```

Fig. 2 Scratch graphical blocks

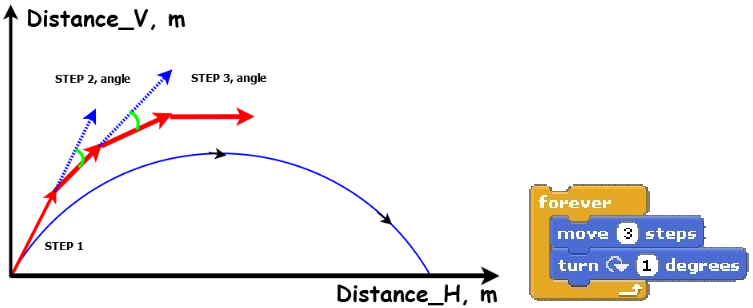
There is less focus on syntax, so one is not required to add semi-colons or square brackets. The blocks are designed to fit together only in ways that make sense, so there are no syntax errors as in traditional programming languages. In this study we show an example of how Scratch can be used to simulate projectile motion in physics. A projectile is any object projected into space by the exertion of a force, i.e. a thrown basketball. In addition to projectile motion, we can simulate other physical phenomena in our immediate surroundings such as the motion of colliding spheres, conservation of momentum and others. Depending on the level of mathematics obtained in secondary school, we can create the simulation with and without algebraic tools. This simulation allows students not only to improve their CT through tinkering but also to focus on physics concepts themselves. Asking students to explain the concept can be preceded by asking them to simulate the projectile motion.

2 Method

Simulation without complex algebra allows us to focus on results - visualization of physical phenomena such as projectile motion. Students can play with scratch code by tinkering to get parabolic trajectory. This can be done by either changing the angle to the horizontal or the vertical distance.

2.1 Changing the angle of launching

Suppose the projectile is launched in a direction defined by an initial angle with respect to the horizontal shown in Figure 3a. This angle can be decreased by a certain amount after each iteration to reach a peak and then ultimately return to the initial y-position. In the example in Figure 3b, the angle is decreased by 1 degrees after every 10 steps. Simulations for three different launch angles are shown in Figure 3.



```
forever
  move 3 steps
  turn ↻ 1 degrees
```

(a) (b)

Fig. 3 Vertical distance change due to change of the launch angle.

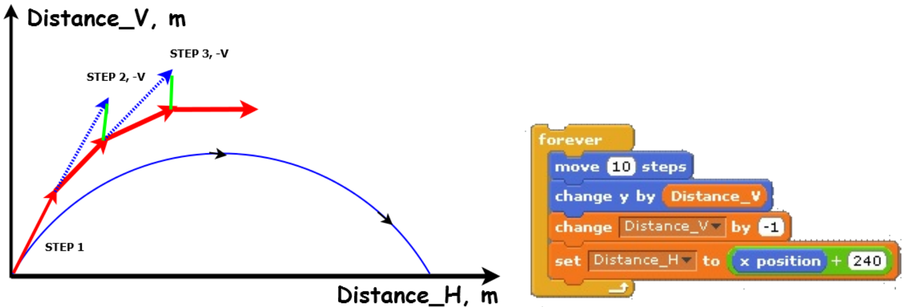
Vertical distance change due to change of the launch angle is

$$Y_{n+1} = Y_o - Y_\theta \tag{1}$$

so distance Y changes incrementally as the angle changes incrementally. The steps are made up of both horizontal and vertical components such that the projectile rise reaches a peak and falls with a trajectory that is symmetrical to the path towards the peak. Calculating the time of flight, the horizontal range, and the height of the projectile can be avoided at this level.

2.2 Changing the vertical displacement

We can also visualize the trajectory by changing the vertical displacement each iteration until it reaches the initial y-position as shown in Figure 4a.



```
forever
  move 10 steps
  change y by Distance_V
  change Distance_V by -1
  set Distance_H to x position + 240
```

(a) (b)

Fig. 4 Vertical distance change due to change of the vertical displacement.

In Figure 4b, the vertical displacement of each iteration, which is made up of 10 steps, is decreased by 1 units. To utilize the entire screen of the scratch interface we set the initial x position to -240 and y position to -166 as (0,0). The iteration stops once a condition of reaching the y position of -166 once again is met.

2.3 Mathematical modelling with algebra

At a higher level of mathematics, algebra can be applied to describe the behavior of a projectile. This can be done in terms of its kinematics motion without dealing with force or energy as a function of time. We assume that the initial and final y-positions of the projectile are the same. We will use the simplest example of a ball launched upwards into the air at an initial angle with respect to the horizontal and velocity. In this case we can calculate: 1. Vertical displacement of the projectile at its peak. 2. Horizontal displacement of the projectile. 3. The launch angle that will result in the largest travel distance. We can then use these scalars and determine the speed and acceleration of the projectile. Visualization of projectile motion helps understanding the dynamics of the motion. Once we visualize the motion, it becomes much easier to answer above questions and calculate them. This visualization using Scratch leads us to understanding the concept of projectile motion in 2D. As a result the following benefits can be gained: 1. Advanced understanding of the concept 2. Advanced problem solving skills. In this simulation, DX and DY changes along the X and Y axis respectively. So using mathematics we can determine these vertical and horizontal changes at different points in time.

$$DY = Vxt \tag{2}$$

$$DY = Vyt + \frac{1}{2}gt^2 \tag{3}$$

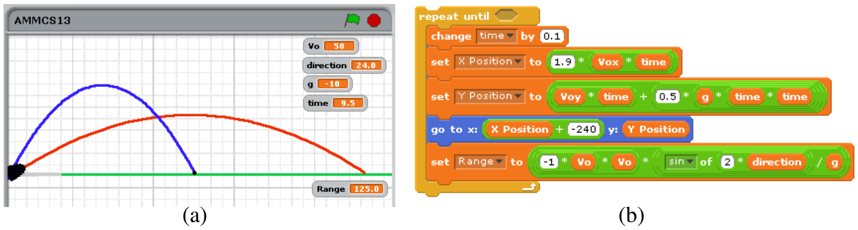
where Vx and Vy are x and y components of velocity respectively. As shown, the horizontal component of velocity is constant and vertical component of velocity is varying with time. The initial velocities can be calculated by multiplying the initial velocity by the cosine or sine of the launch angle

$$Vx = Vcos\theta \tag{4}$$

$$Vy = Vsin\theta \tag{5}$$

It is known that the projectile reaches the highest point when the Vy component of velocity is 0 m/s. The total time of travel is two times the time it takes to reach the peak vertical point. We can also determine the height H from (4) and maximum horizontal distance travelled R from equation (5). Scratch code for this example is shown in Figure 5.

$$R = \frac{V^2 sin2\theta}{g} \tag{6}$$



```
repeat until < >
    change [time] by (0.1)
    set [X Position] to ((1.9) * ((Vox) * (time)))
    set [Y Position] to (((Voy) * (time)) + ((0.5) * ((g) * ((time) * (time)))))
    go to x: ((X Position) + (-240)) y: (Y Position)
    set [Range] to ((((-1) * (Vo)) * ((Vo) * ([sin] of ((2) * (direction))))) / (g))
```

(a) (b)

Fig. 5 Vertical distance change due to change of the vertical displacement.

3 Conclusion

Understanding basic Scratch commands and control tools can help implement more trajectories and generative algorithms by creating and manipulating sequences of graphical commands. Using real life science phenomena as an example, we can create innovative and interactive visualizations to tap into the imagination of students who might never have considered science as fun and playful. Students from various backgrounds tend to be intimidated by the terminology used in science. However, with more exposure to interesting projects, students can start thinking computationally and actively. Computational thinking with a hands-on scratch graphical approach gives them necessary confidence.

References

1. Scratch graphyical platform. -http://scratch.mit.edu.
2. Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, 2012.
3. James Paul Gee. *What Video Games Have to Teach Us About Learning and Literacy. Second Edition: Revised and Updated Edition*. Palgrave Macmillan, 2007.
4. NCWIT. What makes scratch so accessible to novices?, 2008. National Center for Women & Information Technology PROMISING PRACTICES.
5. Seymour Papert. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1):95–123, 1996.
6. Seymour Papert and Cynthia Solomon. Twenty things to do with a computer. 1971. MIT AI Lab memo 248.
7. Alex Ruthmann, Jesse M Heines, Gena R Greher, Paul Laidler, and Charles Saulters II. Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 351–355. ACM, 2010.
8. Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.