Learning Computational Thinking from Rapid Digital Game Creation

Praveen Kuruvada
Computer Science, College of Arts and Science
Oklahoma State University,
Stillwater, OK, USA
praveen.kuruvada@okstate.edu

Daniel A. Asamoah
Information Systems, Spears School of Business
Oklahoma State University,
Stillwater, OK, USA
daniel.asamoah@okstate.edu

Nikunj Dalal
Information Systems, Spear School of Business
Oklahoma State University,
Stillwater, OK, USA
nik.dalal@okstate.edu

Subhash Kak
Computer Science, College of Arts and Science
Oklahoma State University,
Stillwater, OK, USA
subhash.kak@okstate.edu

Abstract--**Computational Thinking (CT) has been described as a universally applicable ability such as reading and writing. In this paper, we describe an innovative pedagogy using Rapid Digital Game Creation (RDGC) for learning CT skills. RDGC involves the rapid building of digital games with high-level software that requires little or no programming knowledge. We analyze how RDGC supports various CT concepts and how it may be mapped to equivalent Java concepts by building the same game using both RDGC and Java. We discuss the potential benefits of this approach for attracting computing majors, as a precursor to learning formal programming languages, for learning domain knowledge, and for bridging the digital divide. We present the implications of this work for teachers and researchers.**

Keywords--**Rapid computer game creation,Computational Thinking, pedagogy, Computing Education, Computer Science curriculum, Information Systems curriculum**

I. INTRODUCTION

Computational Thinking (CT) is an important type of thinking, which combines key elements of analytical, critical, and creative thinking. [8] Seminal article on CT states that "computational thinking represents a universally applicable attitude and skill set that everyone, not just computer scientists, would be eager to learn and use." CT is concerned with conceptualizing, problem-solving and designing systems drawing upon mathematical and engineering thinking using concepts fundamental to computing [8].

However, the mode and method for teaching CT still remains a challenge even though it may be seen as a fundamental skill for problem solving in all disciplines [4]. To this end, the use of Rapid Digital Game Creation (RDGC) has been proposed as a pedagogical framework for teaching CT in an innovative way [2] because video games are attractive and captivating to all groups of people including both adults and children of both genders. RDGC is the process used to build computer games quickly and easily using game creation software that requires little or no programming knowledge. Rapid game creation enables a creator to build a quick prototype game and to see the effects of changes almost immediately [3]. Curricula that have used game design in teaching computing concepts have largely found positive effects on students [10] [11].

In this paper, we describe how RDGC supports the learning of basic and advanced concepts in CT. This paper is organized as follows. In the next section, we describe RDGC and its benefits in teaching CT skills. Next, we discuss how RDGC supports the learning of CT. Using a rapid game development tool called Game Maker, we demonstrate how RDGC could be used as a pedagogical framework to teach CT. In particular, we demonstrate the building of a Pong game and how that can be used to learn CT concepts necessary for computer programming. We map the RDGC implementation constructs to equivalent concepts in the popular Java programming language. We conclude with a discussion on the effectiveness of RDGC in learning CT and discuss implications for future teaching and research.

II. RAPID DIGITAL GAME CREATION

As described earlier, Rapid Computer Game Creation refers to the process of building computer games quickly and easily, using game creation software that requires little or no programming knowledge. RDGC offers an easy and more enjoyable way of achieving this task of building computer games. In an attempt to build a video game, users intrinsically learn basic programming concepts without necessarily realizing that they are using those concepts. Subsequently, when they do learn programming, it is easier for them to understand the programming constructs because they can be correlated with specific examples from the user's own game products.

There are various RDGC tools available such as Game Maker (http://www.yoyogames.com/), Multimedia Fusion (http://www.clickteam.com/website/usa/), Alice (http://www.alice.org/), and Scratch (http://scratch.mit.edu/). For example, Alice is a 3D programming environment used for creating animations for story telling or games [1]. Scratch is also a similar game creation tool used for creating animations, music and art.

The concepts described in this paper are platform independent as any of the tools can be used as a means to enhance CT. For the purpose of this paper, we have chosen Game Maker as the platform because it is popularly available in the public domain and also because of the relatively short learning curve it requires [6]. Besides, Game Maker has got several options such as Sprites (where the user gets the options to create the characters required for the game being designed), Objects (where the characters created in sprites can be imported and assigned with some events), Room (the actual window of the game) and others.

III.   DESIGNING A GAME IN GAMEMAKER

In this section, we take a look at some of the key components to consider in creating a video game using the Game Maker software. We take a game template we designed using Game Maker and draw parallels with the equivalent game designed in Java and subsequently show how it supports CT concepts. We explore how CT as needed for proficiency in a higher level language like Java can be taught to novices in computer programming using the easy-to-use features of a Rapid Computer Game Creation Tool as a precursor to learning the formal language.

Figure 1 shows a screen shot from a prototype Pong game created using Game Maker. The Pong game was designed as a template to illustrate the various programming concepts. The time it takes to create the initial game is as little as 30 minutes. An equivalent Pong game was created using the Java programming language in an effort to understand the correspondence between an RDGC tool and a formal programming language.
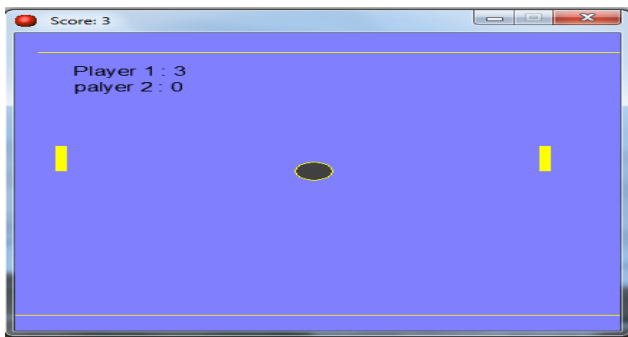

Figure 1. Prototype Pong Game designed using GameMaker

Designing the game in Game Maker requires very little or no programming knowledge. It has several in-built menu options that can be selected as per the user's requirements. Initially we need to create or upload the characters required for the game. This is done in the sprites section where we have an interface where the user could draw a character or upload an existing character.

Once the required sprites are created, we import them to the objects. Once the objects are created, the next step is to assign events to the objects so that they perform actions as required by the game .The use of 'Events' helps the user iterate through different 'if-statements' in trying to make choices as required for the games. At this stage, the user learns what sort of action would be performed upon selection of a particular option. It forms a base to better understand the basic concepts of programming such as the use of sequence, loops, decision structures, and other aspects of programming.

After assigning the required events and setting up other aspects of the game such as the score board and the room design where the objects are to be placed, the final game is ready to be played.
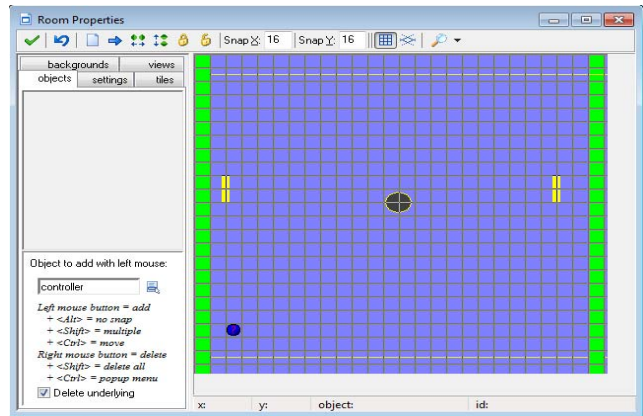

Figure 2. Design of room where the objects are to be placed

IV.   RDGC AND COMPUTATIONAL THINKING

How does designing the game using an RDGC tool help improve CT? To understand this issue, we designed an equivalent Pong game in Java and mapped the explicit programming concepts to various aspects of RDGC. We believe that the use of RDGC by the user before being introduced to formal programming would create a better understanding and improve programming skills [3].

A key aspect of CT involves object-oriented thinking and an understanding of concepts such as objects, events, abstraction polymorphism, and encapsulation. CT also involves the understanding of programming structures such as sequence, decisions, and iterations, All these concepts show how game designing would present a creative approach to learning basic programming concepts. All the concepts and their implementation in the design of the game are explained below.

A.   *Inheritance*

Object-oriented programming allows classes to **inherit** commonly used state and behavior from other classes. Every class inherits by default object classes in the program. The window class inherits the default applet class and the play class inherits JFrames in the game designed using Java.

This method of use of inheritance could be compared to some options in Game Maker which appear similar in action to that of inheritance though they logically do not mean the same. Creation of sprites and inheriting the sprites or their properties and creating objects can be taken as an example for demonstrating inheritance in Game Maker. Table 1 explains how Inheritance concepts are used in the development of the Pong game
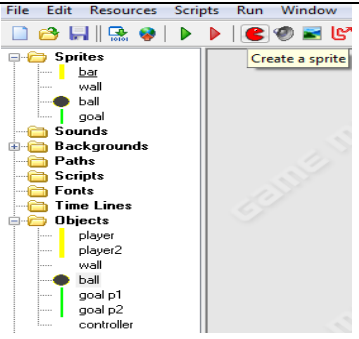
Concept	Sub concepts	Description	RDGC implementation	Java Code
Inheritance	1) *Single Inheritance* 2) *Multiple inheritance* 3) *Multilevel Inheritance* 4) *Hierarchical inheritance*	*Inheritance is the capability of a class to use the properties and methods of another class while adding its own functionality*	 *An Example of the use of inheritance concept in game maker would be the use of objects inheriting the properties of sprites in the pong game.*	*Equivalent Java implementations showing how classes inherit properties form other.* *Class bar extends Object* *{ ....* *}* *Where object is the default in-built class.* *EX:* *Class window extends JPanel* *{ ....* *}*

Table 1. Inheritance concepts in Java

B.   Polymorphism

Polymorphism is a programming language concept that allows values of different data types to be handled using a uniform interface. Polymorphism concepts such as operator overloading are used in the program.

C.   Encapsulation

Encapsulation is a language mechanism for restricting access to some of the object's components. By using access specifiers like public, private and protected, we can restrict access to required methods.

When declared as a public method, the update method can be called by objects in other classes as shown below.

        public void update ()
        {   .....}

Virtual objects are one that are not visible to users, they run in the background but play a important role in the entire process. Example the update functions as sown above.

.   *Use of Java swings and applets*

Java swings are used to create the window where the game is to be played and to add the applet components in the game.

        JFrame f1=new JFrame("game");

The creation of a room in Game Maker, placing various objects in the room and setting up the dimensions of the room can be compared to that of using the swings and applets in Java to create a window for the game.

E.   Graphics

This component in Java is used to create shapes required in the game, paint them and do appropriate changes as required in the game. This is shown in table 3. Creation of sprites, coloring them, setting their dimensions and various other options in Game Maker could be compared to that of using graphics in Java for designing.

Concept	Sub concepts	Description	RDGC implementation	Java Code
Polymorphism	*1) Run Time polymorphism:* *a) virtual function* *2) Compile Time polymorphism* *a) function overloading* *b) operator overloading*	*Polymorphism is a programming language feature that allows values of different data types to be handled using a uniform interface.*	*The two bars in the pong game can be considered as an example for polymorphism since both the bars in the game use the same sprite for their creation but perform different actions in the game.*	*Code shows same method being called using different parameter list.* *Public void setPosition (Point position)* *{ ....}* *public void setPosition(double xps, double yps)* *{ ...... }*

Table 2. Types of polymorphism
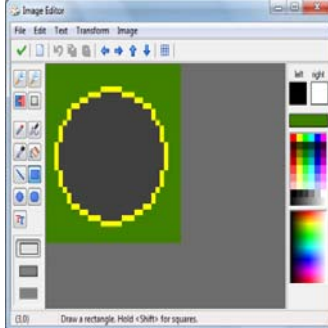
Concept	Sub concepts	Description	RDGC implementation	Java Code
Graphics	*1) 2D graphic API* *2) 3D graphic API*	*There are many in java to enable graphics such as use of AWT (applet window toolkit) or SWINGS* *When the Java applet is activated, Java looks for a method called paint which has a single parameter of type Graphics*	 *Drawing the characters required for the game could be considered as an example for graphics as the design process requires understanding of dimensions, alignment etc.*	*public void paint(Graphics g)* *{* *g.setColor(new Color(0,0,0));* *g.fillOval(x,y,w,h,);* *}* *This will draw t hovel shaped ball* *public void paint(Graphics g)* *{* *g.setColor(new Color(0,0,0));* *bll.paint(g);* *player1.paint(g);* *player2.paint(g);* *}* *This would paint the object created.*

Table 3. Use of graphics in Java and relevant ways of creating graphical objects in Game Maker

F.   Event

Events refer to the significant occurrence or something that takes place upon an action performed by the user. There are various types of events in Java such as window events, key events, and mouse events as stated in table 4. These events allow the user to decide the specific task that is to be performed by the object upon assignment. Adding events to the objects and immediately observing the changes would provide the user a better understanding of the events as used in programming.

G.   Loops and Constructors

The use of basic loop structures such as the If–else structure in the design of the games makes a program perform a specific action or instruction repeatedly till a condition occurs.

As mentioned in table 5, examples of loops include while loop, do-while loop, for loop, if-else ladder and else-if ladder. Use of various events to perform an action repeatedly can be compared to that of loops in a programming language.

A constructor has the same name as that of the class name and used to initialize the variables in the class as the object is created. It is defined with the same name as the class without any return type.
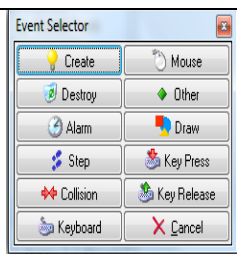
Concept	Sub concepts	Description	RDGC implementation	Java Code
Event Handling	*1) Action Event* *2) Adjustment Event* *3) Component Event* *4) Container Event* *5) Focus Event* *6) Item Event* *7) Key Event* *8) Mouse Event* *9) Paint Event* *10) Window Event*	*Event is an action that is usually initiated outside the scope of a program and that is handled by a piece of code inside the program*	 *The Event Menu shown Above can be considered as an example for the use of different types of events while building a game using game maker.*	*aTimer = new Timer(10, new TimerRepaintListener());* *This event generates a time interval of 10 ms for each update to be performed.* *this.addKeyListener (new barListener());* *This event moves the bars in the pong game by listening to the user actions. This is performed using the key events.*

Table 4. Types of events in Java and ways to assign events to objects in Game Maker

Concept	Sub concepts	Description	RDGC implementation	Java Code
Loops	1) *For loop* 2) *While loop* 3) *Do while loop* 4) *If else loops*	*Loop is a control flow statement that allows code to be executed repeatedly based on a boolean condition.*	*During the process of building a game in Game maker the user goes through many if-else statements for example, the user has to decide when a game would terminate or when a player wins. This process while building a game will make him better understand loops and control flows.*	*if (Condition)* *{ action to be performed }* *else* *{ action to be performed}*

Table 5. Types of loops in Java and similar application in Game Maker

V.   DISCUSSION

In this paper, we have introduced the notion of Rapid Computer Game Creation and discussed ways in which RDGC supports CT. We do ***not*** suggest that game creation is better with RDGC as compared to Java. Rather, as we have argued, when RDGC is used as a precursor to teaching a formal programming language such as Java, there is potential for a student to subsequently gain a better understanding of the programming constructs if the instructor were to explain the constructs in terms of the game that the student has created. We believe that game building, as a pedagogical model helps teach CT in a dynamic manner, which in turn flattens the steep learning curve, needed to learn computer programming.

There is some evidence that game building, as a pedagogical model would be attractive to students even if they are not from the onset, interested in learning about CT concepts [7]. We believe that the CT skills gained through the RDGC tool are an effective way of teaching rudiments of CT to all majors.

Moreover, the gap between those who use information and computer technology and those who do not has widened. To reduce this gap, Rapid Computer Game Creation (RDGC) may be used as a way of introducing students to CT and subsequently bridging the digital divide [2] . Statistically, women, under-represented minorities and the elderly are some of the demographic groups that are on the low end of the digital divide. Since the appeal of games transcends gender, age, and race, introducing RDGC can potentially increase Information Systems (IS) and Computer Science (CS) enrolment among groups historically known to be under-represented in the IS and CS disciplines.

The work reported in this paper raises a number of research issues for pedagogy on aspects related to RDGC. There is a need for empirical studies to understand how well the user can learn CT concepts using RDGC. There is also a need for effective pedagogic models and best practices for the use of RDGC in the classroom.

VI.   CONCLUSION

CS and IS education needs more innovative ways of instruction. Rapid Computer Game Creation has the potential to be an effective pedagogical model in IS and computing courses. It is also a useful pedagogic tool for other academic areas and not just content areas that require computer programming. Game construction and game playing provides more flexibility since it uses a variety of objects and scenarios in an interactive environment. Curriculum designers must consider the inclusion of CT at the pre-college level at par with other fundamental skills such as reading, writing and algebra. Providing students with pre-designed games templates and guiding them to build computer games rapidly constitutes a creative approach for promoting CT and increasing interest in the IS discipline.

REFERENCES

[1]   J .M .Conway , Alice: Easy-to-Learn 3D Scripting for Novices, University of Virginia, 1997

[2]   N. Dala., P. Dalal,, S .Kak,, P.Antonenko. and S.Stansberry. Rapid Digital Game Creation For Broadening Participation In Computing And Fostering Crucial Thinking Skills, Int. J. Social and Humanistic Computing, 1, 2, pp. 123-136, 2009.

[3]   N .Dalal., P .Dalal., S .Kak., Learning Computer Programming with Game Design, International Conference on Computer Supported Education, 2, pp. 135-138, 2009

[4]   M. Guzdial. Paving the Way for Computational Thinking, Communications of the ACM, 51, 2008

[5]   Getting started with scratch v1.4, Lifelong Kindergarten Group, MIT media lab, 2009

[6]   L .Habgood. and M.Overmars, The Game Maker's Apprentice: Game development for beginners. Apress, Berkeley, CA, 2006

[7]   P. Moreno-Ger, D .Burgos, I .Martinez-Ortiz, J.L.Sierr,. and B.Fernandez-Manjon,, Educational Game Design For Online Education, Computers In Human Behavior, 24, 2008

[8]   J. Wing, Computational Thinking. Communications of The ACM, 49, 3, 2006

[9]   YoYo Games, Game Maker, 2010, retrieved January 30, 2010 from www.yoyogames.com.

[10] Bayliss, J. D. and Strout, S. (2006), Games as a "Flavor" of CS1, SIGCSE'06, Houston, Texas, USA.

[11] I. Parberry,.,M. B. Kazemzadeh, and Roden, T. (2006), The Art and Science of Game Programming, SIGCSE'06, Houston, Texas, USA.