A Proposal for Using 3D Technology and Modelling as Teaching Tools in Understanding Computer Programming Concepts

R. Elliott¹, N. Conteh¹, and P. Robertson¹
¹Department of Computer Information Systems, Southern University at New Orleans, New Orleans, La, USA

Abstract - Computer programming involves many complex concepts including requirements analysis, design, implementation, testing. All of these phases lead students deeper and deeper into many different realms of complex computer programming concepts. The aim of this paper is to introduce a proposal to use 3D technology and models to visually model specific programming concepts. Compilation, polymorphism, inheritance and encapsulation will be modelled as 3D images and objects and included in the teaching paradigm. 3D software and hardware such as 3D printers and virtual reality head gear will be used as aids in creating these models and in the presentation of the computer programming concepts. Several groups of students and experiments will be created and the students will be assessed on the comprehension of the selected programming concepts and the results will be analyzed to determine the effectiveness of using this 3D technology.

Keywords: 3D Technology, 3D Modelling, compilation, polymorphism, inheritance, encapsulation

1 Introduction

Teaching computer programming to students with no prior programming experience is a challenge. With all of the different concepts to learn, often present learning hurdles and students don’t excel. This paper presents a proposal in using 3D technology to aid in teaching several challenging computer programming concepts.

1.1 Computer Programming

Computer programming can be defined as the process of creating instructions that will solve some problem in which these instructions can be executed or performed by a computer. There are many computer languages in which these instructions can be written with varying degrees of complexity. Many programming concepts in these languages are often difficult for students to understand quickly and comprehensively. Our research involves using 3D modelling as a teaching tool to aid in presenting specific computer concepts to students to determine if these models will improve the student’s ability to learn the concepts [1]. This research will build upon Henn’s [4] research and test a theory whether or not 3D stereoscopic presentation of our problem domain consisting of computer programming concepts would improve the student’s ability to understand the concepts.

2 Problem Domain: Compilation, Polymorphism, Inheritance, Encapsulation.

The focus area for this research will be in the following four (4) areas: compilation, polymorphism, inheritance and encapsulation. These concepts are often the most difficult programming concepts that beginning students have difficulty understanding and will be the concepts used in this teaching research.

2.1 Learning Styles

According to The Institute of Learning Styles Research [7], there are seven learning styles that may describe how students learn in any discipline. They are known as the seven Perceptual Learning Styles and include Print, Aural, Haptic, Interactive, Kinaesthetic, Olfactory and Visual. This research will focus on the visual learning style in the teaching and learning paradigm.

2.2 3D Technology

According to Ha [3], research findings prove that spatial abilities can be improved significantly through training and 3D tangible models (TMs) and their corresponding computer graphics (CGs) as teaching aids to improve students’ spatial abilities. Our approach is to also use 3D models as a teaching aid in the realm of computer programming to see whether or not students’ ability to understand those concepts improve.

3 Background

This section will provide foundational information on the four concepts under study.

3.1 Compilation

A software program known as a compiler is the major component in the compilation process and must be installed on the computer system for development. The compilation process is highly technical but is a necessity for software engineers and coding specialists to understand. This process
includes using a software or word processing editor or integrated development environment (IDE) software to create a programming solution file. The compiler takes the programming solution, also known as the source file that is written in a computer programming language and converts that source file into machine code which is understood by the host computer system [6].

After an error free compilation process and the machine code file is generated by the compiler, that file is passed to another software program named linker.

The linker program uses the machine code file and resolves all system calls and includes all the necessary pre-coded files from a library of system files many of which are dynamic link library files. It is the linker that produces the final executable program that can be executed and produce the desired programming solutions [5].

### 3.2 Polymorphism

In object-oriented programming, polymorphism (from the Greek meaning "having multiple forms") is the characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are several different kinds of polymorphism [9].

1) A variable with a given name may be allowed to have different forms and the program can determine which form of the variable to use at the time of execution. For example, a variable named USERID may be capable of being either an integer (whole number) or a string of characters (perhaps because the programmer wants to allow a user to enter a user ID as either an employee number - an integer - or with a name - a string of characters). By giving the program a way to distinguish which form is being handled in each case, either kind can be recognized and handled.

2) A named function can also vary depending on the parameters it is given. For example, if given a variable that is an integer, the function chosen would be to seek a match against a list of employee numbers; if the variable were a string, it would seek a match against a list of names. In either case, both functions would be known in the program by the same name. This type of polymorphism is sometimes known as overloading.

### 3.3 Inheritance

Object-oriented programming is a programming paradigm that includes major entities such as classes and objects. These classes are structures also known as superclasses that contain variables and functions or methods which describe the behaviours of the class.

Classes form a blueprint for objects which really are a specialized form of the class. As programmers create classes in their programming solutions, they define variables and methods within those classes. Often, specialized versions of those classes are defined as sub-classes to utilize the predefined data and add new data to make the new sub-class a unique entity. This concept is known as inheritance, sub-classes inheriting properties from a superclass.

### 3.4 Encapsulation

In programming, the process of combining elements to create a new entity is known as encapsulation. Encapsulation is done for the purpose of hiding and or protecting information. In object-oriented programming, encapsulation is an attribute of object design. It means that all of the object's data is contained and hidden in the object. It is restricted and only members of that class can access it [2].

Encapsulation is one of the fundamentals of OOP (object-oriented programming). It refers to the bundling of data with the methods that operate on that data. Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties' direct access to them. Publicly accessible methods are generally provided in the class (so-called getters and setters) to access the values, and other client classes call these methods to retrieve and modify the values within the object.

This mechanism is not unique to object-oriented programming. Implementations of abstract data types, e.g. modules, offer a similar form of encapsulation.

### 3.5 Learning Styles

Learning styles have become very influential within the education field at all levels from kindergarten to graduate school. Today, there is a wealth of material devoted to learning styles in the form of tests and guidebooks and many organizations offer professional development workshops for educators dealing with learning styles. Students have different strengths and preferences in the ways they take in and process information. Proponents of learning style assessment contend that optimal instruction requires diagnosing individuals’ learning style and tailoring instruction accordingly. Typically, these assessments of learning styles ask individuals to evaluate what sort of information presentation they prefer. This research will focus on the student’s Visual modality style [7].

Visual (Spatial) Learning style is a style in which the learner utilizes pictures, images and spatial understanding.

- Learns by using images, pictures, color and other visual media.
- Likes visual stimuli such as pictures, slides, graphs, demonstrations, etc.
- Conjures up the image of a form by seeing it in the “mind’s eye”

### 3.6 3D Technology

3D refers to something that has width, height, and depth. 3D Technology is broad and encompasses many underlying technologies and have been around over one hundred years. A discussion on 3D technology must include how this technology relates to the human experience of 3D.
Humans have 3D vision which is also known as depth perception as each eye forms a 2D image of an object and then our brain processes these two images into a 3D object [mediacollege.com]. There are several tools that people use to gain depth perception as follows [1].

- **Stereoscopic vision**: Two eyes provide slightly separate images; closer objects appear more separated than distant ones.
- **Accommodation**: As you focus on a close or distant object, the lenses in your eyes physically change shape, providing a clue as to how far away the object is.
- **Parallax**: As your head moves from side to side, closer objects appear to move more than distant ones.
- **Size familiarity**: If you know the approximate size of an object, you can tell approximately how far away it is based on how big it looks. Similarly, if you know that two objects are a similar size to each other but one appears larger than the other, you will assume the larger object is closer.
- **Aerial perspective**: Because light is scattered randomly by air, distant objects appear to have less contrast than nearby objects. Distant objects also appear less color-saturated and have a slight color tinge similar to the background (usually blue) [8].

Our research will implement 3D technology to produce 3D computer programming models of: compilation, polymorphism, inheritance and encapsulation. Table 1 lists our hardware and software operating environment.

<table>
<thead>
<tr>
<th>Component</th>
<th>Organization</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Microsoft</td>
<td>Windows 10 64-bit</td>
</tr>
<tr>
<td>3D Software</td>
<td>Sketchup</td>
<td>Sketchup Make v. 17.2</td>
</tr>
<tr>
<td>3D Software</td>
<td>Microsoft</td>
<td>MS Paint 3D</td>
</tr>
<tr>
<td>3D Software</td>
<td>TinkerCad</td>
<td><a href="https://www.tinkercad.com">https://www.tinkercad.com</a></td>
</tr>
<tr>
<td>Hardware</td>
<td>Hewlett Packard</td>
<td>HP Pavilion 17-g121wm</td>
</tr>
<tr>
<td>Hardware</td>
<td>Zeiss</td>
<td>Zeiss VR ONE Plus Virtual Reality Headsets</td>
</tr>
<tr>
<td>Hardware</td>
<td>Pasonomi</td>
<td>3D VR Glasses Virtual Reality Headset for iPh &amp; Android Smartphone</td>
</tr>
<tr>
<td>Hardware</td>
<td>Maker</td>
<td>Select 3D Print v2</td>
</tr>
<tr>
<td>Hardware</td>
<td>Anet</td>
<td>3D Printer</td>
</tr>
</tbody>
</table>

4 Methodology

This section describes the research activities and 3D models that will be used.

4.1 Motivation

As Computer Information Systems professors, the researchers have observed several computing concepts that most of their students have difficulty understanding---compilation, polymorphism, inheritance and encapsulation. We want to incorporate into the classroom something that would help students to quickly and readily understand those concepts. This experiment using 3D programming models is an attempt to determine whether or not using these models will help student comprehension of these concepts.

4.2 Research Question

The research question is as follows.

Will the use of 3D models of compilation, polymorphism, inheritance and encapsulation help students learn and understand the concepts?

4.3 Hypothesis

Our hypothesis are as follows.

Null Hypothesis: Using 3D CIS programming models will not improve student test scores by an average of 20%.

Hypothesis: Using 3D CIS programming models will improve student test scores by an average of 20%.

4.4 Experiment

This research will use an experiment and will involve randomly dividing our students (subjects) into a control group and an experiment group depending upon class size. The experiment will consist of a quiz on the four (4) areas: compilation, polymorphism, inheritance, and encapsulation.

The control group will only receive classroom instruction using traditional textbook and computer information providing the content.

The experiment group will be paired with the researchers and will receive classroom instruction using traditional textbook and computer information providing the content and will use 3D models as learning and teaching tools.

The four (4) 3D models will be created using the 3D application software and hardware to model the following concepts: compilation, polymorphism, inheritance, and encapsulation. These models are a very important key in the research and will take much thought in design and creation. We need the 3D hardware and software necessary to build all four (4) models.

These models will be our independent variables and made available to the experiment group but not to the control group. Both groups will be instructed in the four (4) areas: compilation, polymorphism, inheritance, and encapsulation. Both groups will be given a quiz in the four areas listed above and the test results will be analyzed and conclusions will be drawn.
5 3D Models-Compilation, Polymorphism, Inheritance, and Encapsulation

The following are the tentative 3D prototypes of each of the programming concepts under study. Our theory in focus will be on visual learning.

5.1 Compilation Prototype Model

The chosen image to model compilation will be a simple multi-layered pill as depicted in Figure 1. There will be a total of six (6) components modelled to form a pill while describing the processes involved in program compilation. They all fit neatly into one visual pill. The outermost pill being the editor. The model dimensions will be 12 x 6 x 6 (length, height, depth) in inches.

Figure 1

5.2 Polymorphism Prototype Model

Polymorphism in object-oriented programming involves allowing objects to take on different forms with different behaviour. In Figure 2 below, the mouse class is the super class and there are three (3) derived classes listed. The three (3) derived classes all have a method named “connectionType” that performs differently. In this example, “connectionType” is exhibiting polymorphic behaviour. The same method name, but with a different behaviour.

Figure 2

The mice listed in Figure 3 above will be the 3D objects modelled to teach the concept of polymorphic behaviour: a USB mouse, a serial mouse and a wireless mouse respectively. They all perform the same functions and tasks, however, they are implemented in the computing environment very differently.

Figure 3

5.3 Inheritance Prototype Model

Inheritance is about automatically passing on attributes from one object to another. Researchers will create a model of a seed to teach the concept of inheritance. The seed contains all of the attributes of the fruit that eventually automatically will be produced. The 3D seed model as listed in Figure 4 will contain the core components that will always produce a red delicious apple. A 3D seed will be modelled to convey the concept of inheritance as all fruit inherit from a seed, all methods and variables are inherited from a super class.

Figure 4

5.4 Encapsulation Prototype Model

The concept of encapsulation will be modelled as a 3D baseball. Initially, an initial sphere will be created to depict the inner core of a baseball. Which will be protected by two intertwined covers. Thirdly, that single object will be protected by the white leather cover. The essence of encapsulation is to protect or hide components from user access. The real baseball has two protective or encapsulating objects.

Figure 5

6 Data Analysis and Results

This section will contain all artifacts used in the experiment.

6.1 Data Analysis

Surveys will be provided to the groups to collect data items: sex, age, race, classification, major, gpa, quiz score. Statistical analysis will be done on the collected data.
6.2 Results
The results of grading the quizzes and providing reports by the different data items listed above.

7 Conclusions
This section will discuss the researcher’s findings and future work.

7.1 Dependencies
This research is dependent upon the researchers successfully writing a grant proposal for funding of this research project. The research question and the hypothesis will be answered upon completion of:
- acquiring the 3D hardware necessary to build the models.
- researchers learning and using the 3D software to be used in conjunction with the 3D hardware

7.2 Threats to Validity
The following are the threats to this study including the software tools that were chosen.
Application Software. The application software of this experiment will be programming and computer science oriented. These types of application software are too specific in functionality to make a general statement about successfully using other types of application software.
Operational Environment. The quizzes and operating system applications will be written in and executed within the same Windows operational environment. One operational environment is not adequate to make a generalized statement about successfully conducting this research in another type of operational environment.
Software Tools. The major software tools used in conducting this research will be Windows 10, Microsoft 3D Paint and Sketchup. Selection of these tools restrict making generalizations about the outcome of successfully conducting this research if other software tool choices are selected.
Verification Tasks. The process of verifying software requirement characteristics will be only executed in one experiment. One application of the verification tasks is not enough to make a generalized statement about other types of verification.

8 References