The Use of MUPPETS in an Introductory Java Programming Course

Kevin J Bierre
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623-5608
585-475-5358
kjb@it.rit.edu

Andrew M Phelps
Rochester Institute of Technology
102 Lomb Memorial Drive
Rochester, NY 14623-5608
585-475-6758
amp@it.rit.edu

ABSTRACT

"The Multi-User Programming Pedagogy for Enhancing Traditional Study" (MUPPETS) system has been under development at RIT for the last three years. This multi-user environment is designed to allow students to develop visible 3D objects in Java within a game-world environment with minimal knowledge of graphics programming. Students can interact with these objects through an interface built into the system. (Technical aspects of the MUPPETS system were previously published by the authors at CITC4) [1].

In testing the usefulness of MUPPETS as a teaching tool, we have developed a series of course modules that use the environment as its programming environment. The existing "Programming for Information Technology III" course is the ideal place to perform an initial test of this nature, as students have some base familiarity with the Java language but have not yet completed their undergraduate programming core. Students in this course have a final group programming project that we intend to use as the initial test, and develop further MUPPETS modules downwards towards the initial freshman experience.

In the past students used a package called "Robocode", which is available from IBM [2]. This project involved programming a virtual robot that could "fight" in an arena according to some agreed upon set of rules, which were developed both as part of the Robocode package and discussed and agreed upon in lecture. While the students enjoyed this project, the proliferation of available code on the Internet for the framework led to this project being removed from the course. We have implemented a variant of "RoboCode" in MUPPETS that addresses the code availability issue and provides a more interesting and graphically rich environment for the students.

This paper shall discuss the reasons for the implementation, what we expect the students will gain from the use of MUPPETS based project, and possible methods of comparing this approach to the methods previously used in this course. Also discussed are additions to the MUPPETS system made to facilitate its classroom use including a re-implementation of the Swing graphics classes such that 2D interfaces are available in 3D, and model loading and texturing tools that allow custom robot creation and customization.

Categories and Subject Descriptors

K. Computing Milieu

K.3 Computers and Education

K.3.2 Computer Uses in Education

Subject Descriptor: Collaborative Learning.

General Terms

Experimentation, Human Factors, Languages, Theory.

Keywords

Game Programming, Programming Education, Virtual Worlds, Graphics.

1. INTRODUCTION

Many students find learning programming to be a difficult and unpleasant task. There are a variety of possible reasons for this perception, such as a lack of motivation about the subject matter, a lack of prerequisite skills such as problem solving, or just the reputation of the course. [3]

The Information Technology department at the Rochester Institute of Technology has been attempting to make the subject matter more accessible to students through a variety of changes. Students can proceed through our introductory courses in either three or four quarters [4]. The longer sequence allows the students to concentrate more effort on areas that are known to be troublesome, such as object-oriented concepts. We are also investigating the use of cohorts of students to see if that improves retention within the program.

In addition to the above changes, we are planning to introduce a collaborative virtual environment (CVE) called MUPPETS into the pedagogy of how the courses are delivered. The initial use of MUPPETS will be in the last introductory programming course as part of the final project that is an integral part of the curriculum. As more modules and resources are developed, and more testing can take place, we plan to introduce MUPPETS earlier in the course sequence, allowing the students to become familiar with it well in advance of the final project.

2. CURRENT ISSUES

As part of the final programming course, students spend last 2-3 weeks of the quarter working on a group programming project. For many students, this is their initial introduction to working on a programming team, so care is taken to cover topics such as design documentation, project planning, creating test plans, and other aspects of team-oriented programming.

In the past, students created various types of client-server systems, including chat rooms, small multi-user logic games, and simplistic studies of encryption. While many found this moderately interesting, there was not a large degree of enthusiasm about the topics provided. However, when we located a project called "Robocode" through the IBM Alphaworks site, this seemed to finally catch their interest.

Robocode is a Java based API that allows students to create robots that can then be used to fight one another in virtual 2-dimensional arena. Students had to design their robots using the API methods. In order to do this successfully, they needed a solid understanding of various parts of the language.

Robots were tested in a tournament on the last day of class. In addition, students had to deliver a presentation on their design and what they learned from the experience. Overall, the students appeared to have enjoyed the project and it encouraged them to explore areas of the Java language that were not presented in class.

The first quarter that Robocode was used was deemed very successful. Unfortunately, later quarters did not work out as well. The major problem was the increasing availability of Robocode code on the Internet. It was possible for students to create a robot by taking bits of other robots that were available online and patching them together to form a new one. It was difficult to tell what code a group developed and what code came from other sources.

Students also had issues with the user interface. Robocode's interface is a top view of the arena and does not have any sound capability. There is no way to alter the view. We suspect that rising expectations of a student population familiar with current computer and video games was the source of the complaints.

Finally, we noted that the victorious robots tended to be "wall huggers". (Meaning that they moved to the outer wall and spent their time circling the arena.) This meant that a poor choice of tactics early in the design phase could affect a teams overall performance, and that there was seemingly an obvious "best strategy", thus discouraging many groups from trying several approaches to the problem.

A proposed solution to the problem was to develop our own package for the final project, similar in some respects to the Robocode system, but also highly customizable and with advanced capabilities. By basing it on MUPPETS, we addressed the above issues easily:

- With our own package, we could remove the code reuse issue by updating the package with different behaviors between quarters. Code that was working in one quarter could be invalid the next.
- MUPPETS has the three dimensional views that students are used to seeing in professional quality video games. In fact, it has been used to drive virtual reality displays and a large number of input devices, including game controllers and other haptic interfaces. It also contains spatial sound capability.
- By having control of the package, we can make modifications to avoid the appearance of an overwhelmingly successful strategy. Indeed, the entire arena could be changed such that strategies that were largely successful one quarter would be easily beaten in future matches.

3. RECENT RESEARCH IN CVE's

Over the past few years, a variety of research has been done on various types of collaborative virtual environments to determine their effect when used to teach programming.

In one case, the LambdaMOO software originally developed by Xerox PARC was used to create a text based interactive environment. Course material was introduced through lectures. Students would work within the MOO environment on assignments. They would be able to communicate with other students, as well as the instructor. At the end of the five week experimental period, student's knowledge was assessed through an examination. [5] Student reported that the virtual environment helped them learn the material. The examination results appear to bear this out.

In two cases, Lego Mindstorms robots were used to teach programming. One case relied on the physical robots as the basis for learning to program. [6] In the other case, a simulator was provided and student work was initially tested on the simulator prior to being added to the robot. [7] While these methods appear to have met with some success, they are not true CVE's . (An interesting aspect of use of the physical robots was the discovery by the students that plans that looked good on the screen and performed well on the simulator often did not hold up well when moved to a real environment.)

Karel the Robot has been used to teach programming since the 1980's. An update of the original ideas was produced to introduce object oriented programming. [8] Unfortunately, it used a proprietary language. The authors of that system gave permission for a group to create a Java version of Karel++, using the same type of virtual world the text describes. [9] This group also took five weeks to introduce basic object oriented concepts to introductory students. While this study did not perform a formal evaluation of student performance, they felt the students learned the material and had fun in the process. The visual component of the robot environment was mentioned as a major feature in gaining the student's interest in the topics presented.

The PUPPET project was created to investigate methods to encourage reflection on what a student has recently experienced. In this study, children were placed in an environment where they interacted with characters. The children were able to reflect on what they had just experienced within the CVE and draw conclusions. The students found the environment engaging and had no problems determining how to interact. This seems to indicate that this type of environment could be used for teaching at an elementary or secondary school level. [10]

4. THE MUPPETS ENVIRONMENT

MUPPETS provides students a three-dimensional view of the environment similar to that found in commercial grade video games, coupled with a command console and a Java IDE. These are the three primary modes that students use when working within MUPPETS. Each is described in detail below:

4.1 Normal View

The view first seen by a student when they enter MUPPETS is shown in Figure 1. They are represented by an avatar and have a camera that they can adjust to control the view they can see. The field of vision is about 60 degrees, which is less than actual field of the human eye.

Under the regular version of MUPPETS, the student can move their avatar around and interact with other objects that they create or find within the environment. This is done using keyboard commands.

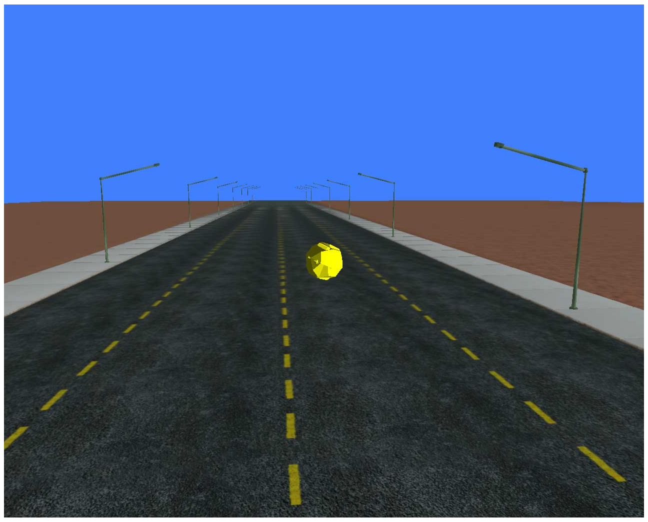
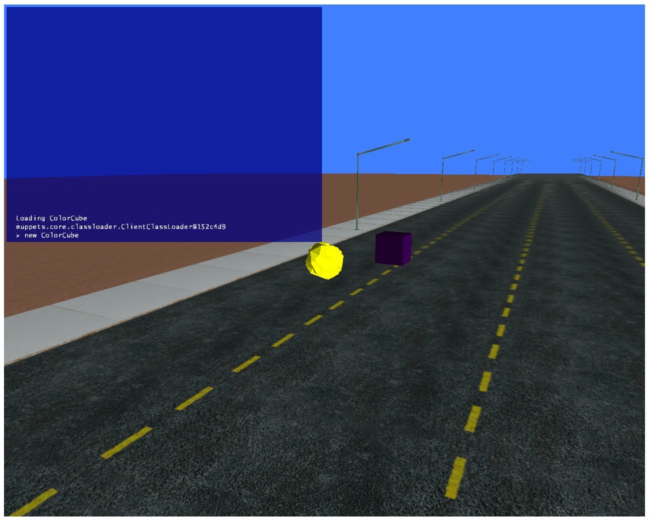


Figure 1. Initial view of MUPPETS environment

The version that is being used for the initial test is a bit different, because the student is "along for the ride". The robot they create has a preprogrammed set of actions and the student has no real control over its actions once the robot enters the arena.

4.2 The Console

The user can issue commands to MUPPETS through the console. This allows keyboard input to affect the system. There are a variety of commands that can be issued, although the primary ones involve the creation and removal of objects. A key binding mechanism is available that will allow commands to be mapped to different keys, removing the need to bring the console up for common operations


```
Loading ColorCube
muppets.core.classloader.ClientClassLoader@152c4d9
> new ColorCube
```
Figure 2. MUPPETS Environment with console displayed

4.3 The IDE

MUPPETS allows the student to create Java classes using a built in integrated development environment. A student can select a class to use as a parent to develop a new type of object. (All classes must implement the Muppets interface) Or they can alter the behavior of an existing class. The first step is to select a class from the screen showing the possible options. (Figure 3)

Once a class has been selected, the student is placed in an editor. From the editor, they can alter the code, save it on the local machine, and compile. Compilation errors appear on the upper part of the screen.

If the compile was successful, the student can immediately create a new instance of the object and observe its behavior. This allows for rapid development and debugging.

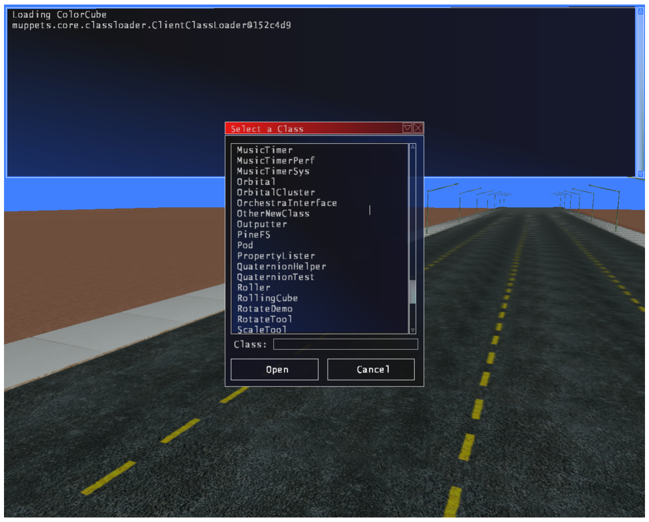
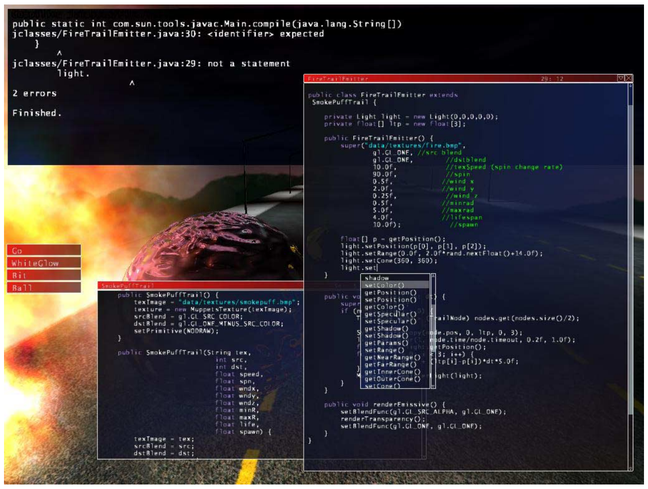

```
Loading ColorCube
muppets.core.classloader.ClientClassLoader@152c4d9
```

Figure 3. MUPPETS class selection dialog



```
public static int com.sun.tools.javac.Main.compile(java.lang.String[])
jclasses/FireTrailEmitter.java:30: <identifier> expected
    }
        ^
jclasses/FireTrailEmitter.java:29: not a statement
        light.
                     ^
2 errors

Finished.
```

```
public class FireTrailEmitter extends
 SmokePuffTrail {

    private Light light = new Light(0,0,0,0,0);
    private float[] ltp = new float[3];

    public FireTrailEmitter() {
        super("data/textures/fire.bmp",
                gl.GL_ONE, //src blend
                gl.GL_ONE,          //dstblend
                10.0f,              //texSpeed (spin change rate)
                90.0f,              //spin
                0.5f,              //wind x
                2.0f,              //wind y
                0.25f,              //wind z
                0.5f,              //minrad
                5.0f,              //maxrad
                4.0f,              //lifespan
                10.0f);              //spawn

        float[] p = getPosition();
        light.setPosition(p[0], p[1], p[2]);
        light.setRange(0.0f, 2.0f*rand.nextFloat()+14.0f);
        light.setCone(360, 360);
        light.set
    }

    public vo                 ) {
        super
        if (n
            T                ailNode) nodes.get(nodes.size()/2);

            S                de.pos, 0, ltp, 0, 3);
            l                de.time/node.timeout, 0.2f, 1.0f);
            f                tPosition();
            f                3; i++) {
                             tp[i]-p[i])*dt*5.0f;
            }
            W                ght(light);
        }
    }

    public void renderEmissive() {
        setBlendFunc(gl.GL_SRC_ALPHA, gl.GL_ONE);
        renderTransparency();
        setBlendFunc(gl.GL_ONE, gl.GL_ONE);
    }
}
```

```
    public SmokePuffTrail() {
        texImage = "data/textures/smokepuff.bmp";
        texture = new MuppetsTexture(texImage);
        srcBlend = gl.GL_SRC_COLOR;
        dstBlend = gl.GL_ONE_MINUS_SRC_COLOR;
        setPrimitive(NODRAW);
    }

    public SmokePuffTrail(String tex,
                          int src,
                          int dst,
                          float speed,
                          float spn,
                          float wndx,
                          float wndy,
                          float wndz,
                          float minR,
                          float maxR,
                          float life,
                          float spawn) {
        texImage = tex;
        srcBlend = src;
        dstBlend = dst;
```

Figure 4. MUPPETS IDE code editing screen

5. THE PROJECT

For the final project in the introductory programming sequence, students create robots that will be used to “fight” in an arena. This strategy was well received with the previous Robocode project in terms of building class camaraderie as well as motivating students through peer-pressure rather than grade-oriented measures. Note that the term “fight” here refers only to robot vs. robot competition: several winning strategies did not revolve around violence to the other robot, but rather clever manipulation of the surrounding environment.

The arena in which the “fight” takes place is a bounded space in three dimensions. One of the problems we noted is the earlier forms of this project was tendency for successful robots to hug the outer wall. We eliminated that tactic by removing the outer wall. In effect the arena wraps around so that robots exiting on one side appear on the corresponding opposite side. (That behavior can be altered if it turns out students develop tactics relying on the arena’s behavior, from quarter to quarter to prevent code reuse.) The arena is flat and contains no obstacles aside from the other robot. For more complex battles, obstacles can easily be added, and simple collision detection and avoidance routines are already available in the base package.

Robots consist of a platform with a fixed laser weapon pointing forward. The platform also contains a sonar unit. The students can control the movement of the robot, the use of sonar to locate the other robot, and the firing of the laser. This allows for a fairly broad range of behavior: some strategies focus on movement and dodging, others on continual fire and random sweeps of the area. In almost all cases, it is likely that more than one strategy will be successful, and that several have the possibility to win the competition.

An interface is provided to students that describe the methods they may use to program the robot. These methods were developed to use existing MUPPETS code that requires a much more detailed knowledge of the software and that is likely beyond the typical introductory student. Thus, by creating a set interface, or group of base functions for the students to work with in their own classes, they are shielded from most if not all of the complexity of drawing objects in three dimensions and dealing with graphics programming in general. In fact, the core of the graphics system is not even written in Java, it relies instead on a complex integration of Java and C/C++ across the JNI [11]

The entire MUPPETS environment that the students use is provide on a server all students can access through their departmental account. Currently only the Windows and Linux platforms are supported on the X86 32-bit architecture, but plans to support other operating systems and hardware platforms are currently being explored. All of the interface documentation is available using the standard Java documentation format. In addition, students are provided with the following documentation:

- Instructions on how to download and install MUPPETS on a PC
- Instruction on how to start MUPPETS and use the standard set of commands.
- Instructions on how to customize MUPPETS commands.
- An overview of what MUPPETS is and how it works, in terms a new programmer can follow.
- Instructions on the use of the integrated development environment.
- Documentation on the interface we use to program the robots, with simple examples.

Sample robots are provided for the students to examine. This also provides some opponents for use during testing. A screenshot of a default tank implementation is presented in Figures 5A and 5B.

Figure 5A. Typical tank implementation from an aerial perspective.

6. PROJECT CONSIDERATIONS

There are several goals that we try to complete as part of this project. We feel that each of these areas are an important part of the introductory programming experience.

6.1 Teamwork

Prior to starting this project, all programming assignments had been individual work. It was deemed critical for students to be

Figure 5B.Close-up of a student created tank mesh.

exposed to the type of teamwork that would most likely be required during their co-op blocks, and eventual induction into the workforce.

Students are generally allowed to select their teammates, although there have been cases where the instructor has altered the composition of teams for various reasons. (For example, teams made up entirely of students who have exhibited weak programming skills). We have found that two person teams seem to work best for this project. Three person teams lead to coding conflicts, since there is only one major block of code that is used for the project, and not usually enough distinctly separate tasks that they can be divided across a larger team.

Another critical part of teamwork is peer evaluations. Each team member evaluates themselves and their teammate. They look at how much each contributed, how reliable they were, and how much effort they put out. Additional comments can also be added to the evaluation sheets.

6.2 Project Planning

This is also the first assignment that demands planning for successful execution. The team is responsible for several deliverables, including design notes, a journal, a presentation of a prototype idea, and of course the final demonstration and presentation. Some planning as a group is required to ensure all requirements are met completely.

6.3 Testing

We have emphasized testing for the typical assignment. In this project it becomes more critical, since the robot is expected to compete in an arena in front of the entire class. This form of motivation seems to increase the student's interest in providing a fully tested product, if for no other reason than to avoid embarrassment during the trials.

6.4 Presentation Skills

As a group, programmers are not generally known for their presentation skills. Unfortunately, it is a fact of the industry that a programmer will be called upon to present their work to others. It is important for students to have this initial experience. Faculty prepare the students by going over basic presentation skills, including specifics on presenting technical material such as programming designs. As part of the requirements, all team members are expected to participate in the presentation.

7. EXPECTED RESULTS

Students are expected to be able to demonstrate the following skills at the end of the project:

- The ability to use a Java package and its associated documentation to create a functioning program.
- The ability to define tasks, create a simple time line and assign work to the members of the team.
- The ability to track their progress through the use of design documents and a project journal.
- The ability to plan and deliver a presentation describing important aspects of their project.

8. ASSESSMENT OF RESULTS

One of the major questions that need to be answered is whether this project has increased the student's knowledge of programming and has met the expected results listed in section 7.

Much of this knowledge can be gained through the use of the presentation. The presentation for this course is assigned by providing the students with a set of questions to be answered. Usually these are basic project post mortem project deconstructions as seen in current literature. By adding additional question regarding the expected results listed above, we should be able to determine which goals have been met and which may need to be adjusted. In addition, receiving the design documents and journal from each team will provide some insight about how effective their planning techniques were when creating the program.

9. FUTURE WORK

Since this project is first being tested during the 2004-2005 academic year, we expect to make some changes to the project as results become available.

One potential area of change would be in the arena. The addition of obstacles would increase the complexity of the problem. This could take the form of adding separate objects, such as houses or wrecked robots to provide cover. Adding hills and valleys could also alter to the terrain and provide several nuances in terms of strategic advantage and robot AI. Changes to the way the arena wraps around would also affect play.

Altering the robots themselves could also change the way the game is played. Changes such as limiting the detection range or changing the weapon used will have a major effect on the program and the strategies incorporated for victory.

10. ACKNOWLEDGMENTS

The authors would like to thank RIT's Provost's Learning Initiative Grants programs for providing the initial funding for the MUPPETS project, as well as Sun Microsystems for matching educational grants that made possible the initial development of the system. The authors would also like to thank Jeffrey Sonstein and William Stratton of the Information Technology Department at RIT for their support in the project and their work in the initial grant application, as well as Eydie Lawson, former Chair of Information Technology for her continued support. A very special thanks must be given to Ed Huyer, who implemented much of the Robocode environment within MUPPETS, as well as to Dave Parks and the MUPPETS team for their continued hard work in developing the system and its capabilities. More information and samples of the MUPPETS environment can be obtained at http://muppets.rit.edu.

11. REFERENCES

[1] Phelps, A, Bierre, K, and Parks,D, *MUPPETS: multi-user programming pedagogy for enhancing traditional study,* Proceeding of the 4th conference on Information technology education , October 2003, Lafayette, Indiana, USA, 100-105

[2] Nelson, M. Robocode: Online: http://robocode.alphaworks.ibm.com/home/home.html

[3] Jenkins, T. *On the Difficulty of Learning to Program*. 3rd Annual LTSN-ICS Conference, Loughborough University, Leicestershire, UK, 2002, 53-58

[4] Whittington, K, Bills, D, and Hill, L, *Implementation of alternative pacing in an introductory programming sequence*, Proceeding of the 4th conference on Information technology education , October 2003, Lafayette, Indiana, USA, 47-53

[5] Towell, J and Towell, E, *Reality Abstraction and OO Pedagogy: Results from 5 Weeks in Virtual Reality*, OOPSLA '03 October 26-30, 2003, Anaheim, California, 162-165

[6] Barnes, D, *Teaching Introductory Java through LEGO MINDSTORMS Models*, SIGCSE '02, Feb 27 – Mar 3, 2002, Covington, Kentucky, USA 147-151

[7] Flowers, T, and Gossett, K, *Teaching Problem Solving, Computing, and Information Technology with Robots*, Journal of Computing in Small Colleges, 2002 17,2, pg 45 – 55

[8] Bergin, J, Stehlik, M, Roberts, J, and Pattis, R, *Karel++: A Gentile Introduction to the Art of Object-Oriented Programming*, John Wiley & Sons, 1997

[9] Becker, B, *Teaching CS1 with Karel the Robot in Java*, SIGCSE 2001, Feb 2001, Charlotte, NC, USA, 50-54

[10] Marshall, P, Rogers, Y, and Scaife, M, *PUPPET: playing and learning in a virtual world,* http://www.cogs.susx.ac.uk/interact/papers/pdfs/Playing%20and%20Learning/Tangibles%20and%20virtual%20environments/Marshall_IJCEELL.pdf

[11] Phelps, A and Parks, D, *Fun and Games: Multi-Language Development,* Queue, Vol1,10, Feb 2003, 46-56