Course Title: SDE 130 Introduction to Object Oriented Programming

Course Leader: David Maruszewski

Expected Learning Outcomes for Course

- Evaluate and understand the benefits and challenges associated with an object-oriented analysis design approach to software and project development
- Identify the key concepts used in object-oriented development including inheritance, encapsulation, data types, control flow, polymorphism and programming techniques
- Identify design patterns in terms of participating objects and classes and the roles they take on relative to the problem design solves
- Investigate and evaluate OOA&D (object-orientated analysis and design) tools, methods, and models that are available and currently used in business practice
- Design an algorithmic, object-oriented solution that meets the specification of a programming problem
- Document OOD (object-oriented design) diagrams that meet industry standards
- Adeptly model and animate in 2 dimensions and 3 dimensions
- Logically formulate scripts and/or programs to solve problems
- Understand and articulate interactivity in the gaming industry, including the connectivity between computer art and programming
- Apply programming and artistic theory in practical applications
- Demonstrate problem solving skills through verbal and written media
- Apply rudimentary physical principles to animations or simulations

Assessment

(How do students demonstrate achievement of these outcomes?)

Students are required to complete a final project which is applied against a grade sheet. This project was created to test skills gained throughout the course. Almost all the sections map to at least one outcome which are compiled and reported in Moodle.

Three exams are issued to help confirm the findings of the project grade. Questions pertaining to classes, events, and math application are evaluated.

Validation

(What methods are used to validate your assessment?)

Currently, all grades sheets are held for two semesters and composite data is used to show trends. Certain chosen questions on exams should help verify or contradict findings.

Results

1. Math applications are an issue. High school math is all that is needed for this course, but the application of that math is hard for the students to grasp.
2. Switching to C# has improved understanding of coding theory. However, less material was covered.
3. Flowcharts and pseudocode were taught and stressed to the point where it was vocally expressed that they would be on every test. Students refused to practice these on their own and almost all students left one or both out on the project. The grade sheet for the

project was given to the student at the beginning of the semester, such that the students knew they would be graded on it as well as it was explained on the project description.

4. Improvements were seen in the area of professionalism and documentation (34% to 56%)
5. Oddly enough, our best outcome had to do with more of a design outcome. (Students will demonstrate layout and composition in their pieces through the use of balance, hierarchy, emphasis, unity, movement, contrast, rhythm, focus, use of grids and white space: 88%)
6. "Understand and articulate interactivity in the gaming industry, including the connectivity between computer art and programming" outcome improved from 69% to an 81%.

Follow-up

(How have you used the data to improve student learning?)

1. This continues to be a problem. Unfortunately, high schools are not set up to close this gap and college standards are not set to fix the problem either. I must re-teach students 9th grade level math. The outcome with the lowest score (Students will demonstrate problem solving skills by analyzing, selecting and applying tools appropriate for a specific solution). This did improve from a 21% to a 67%. However, this was an online class where I distinctly put in these math operation into videos they had to watch. Next semester, we will see if this keeps when the on-campus course is executed.
2. The online delivery was better as I could fit exactly what I needed in videos. Implementation in the on-campus course will be re-worked to have the same feel.
3. The modern student tends to do what they like to do, and try to forgo the less enjoyable. They hope that good work on what they enjoy will make up or even erase uncompleted material. Unfortunately, this being an introduction course has many of these students not understanding this won't work. The lesson that they learn will pass on to the next class but is hard to fix for this course. I did replace the rubric on the project with a more understandable one.
4. This stayed at 50% this semester.
5. This stayed high at a 95%. I thought at this was because the class is a mix of designers and programmers. As it turns out, programmers like to be creative, too. This class allows them to vent their design and creativity in a way that other of their analytical class usually don't. I'm going to take this a a good thing.
6. Focusing on objects and classes in project helped aid this. The project seems to have the biggest impact. This improved again to a 97%. I believe the current methods of assigning grades and content should be maintained.

Budget Justification

(What resources are necessary to improve student learning?)

Microsoft Visual studio needs to be used for this course now. So, we either need it purchased or use Thawspace to download the free versions.