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
- Indentify 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. The sections pertaining to Flowcharts, Storyboards, “two player” and “piece” are evaluated.

Three exams are issued to help confirm the findings of the project grade. Questions pertaining to loops, methods, 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. Computer concepts were easily grasped when using Alice. Although, students may not be able to articulate them well. They are understood, but not defined by the students.
3. Alice used to be used and many concepts were forged through in a short period of time. Although, mastery was not necessarily achieved. Switching to C# has improved understanding of coding theory. However, less material was covered.

4. Using many rules in a common animation confuses students and their programming suffers.
5. Using outside sources has fueled students project/code work.

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

1. I have to re-teach some math in this course. I have spoken with some high school officials at advisory boards in hopes to improve this.
2. We switched from Alice to C# and Microsoft Visual Studio.
3. This may be an effect of a first time run material. However, a more comprehensive lesson plan may be in order to get through concepts especially involving OOP and OOD.
4. Moving to event driven programming in conjunction with OOP has helped this. Now animation and programming can be compartmentalized if needed.
5. This is good and they even referenced well.

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 bought or Thawspace to put the free version of it.