UC San Diego Extension, 0176S
La Jolla, CA 92093-0176

Effective Start Term: SPRING 2016

COURSE OVERVIEW

Course Title: **Level 2: Build Your Own Game** **Course Number**:

Number of Units:

Total Hours of Instruction: 25 - 35 (depending on complexity)

Course Description

During this course, students learn to design and build a game that aligns their interests (e.g. airplane piloting, spearfishing, drumming, math) with their own creativity, to create an original game.

This is an exciting time for students to to apply their programming skills to create their first real-world application, with heavy emphasis on the end user as a driver for the evolution of the game. The functional game developed in this course goes through multiple iterations of usability testing and development. The projects are publicly published on GitHub and will form part of the student's professional portfolio of work.

The purpose of this course is to flex the programming skills acquired in the previous two levels, to publish a significant piece of the students' professional portfolio, and to give the student the confidence and motivation to create, rather than just consume, the digital world around them.

Content and Evaluation

Implementation of the project is completed by the student with the guidance of a professional software engineer who has at least 3 years of software industry experience. Through lectures and project-specific examples, the software engineer mentors the student in Agile/Lean methods, and industry best practices such as object oriented design and refactoring.

The User eXperience (UX) element of this course has been designed by President of San Diego Experience Design Professionals Network, Paul Hong, specifically for League students.

This student is evaluated for this course based the following:

- Demonstrated understanding of Object Oriented Design (OOD) through use of at least 3 classes.
- Application is published and documented on GitHub.com in compliance with League GitHub guidelines.
- Completed usability testing log used to capture the specifics of at least 5 usability testing sessions.
- Deliver a short presentation to explain project challenges, technical implementation, and now the project was adapted in light of user feedback
- Project must be presented and demonstrated to League Lead Teacher, League Executive Director and at least one parent on final Demo Day.

Extra Credit

n/a

Methods of Instruction

X	Class Discussion/Discussion Boards	X	Lectures
☐	Field Trips	☐	Case Studies
☐	Group Work	X	Other: Programming Assignments

Out of Class Assignments

Total hours expected to complete assignments: n/a

☐	Textbook Exercises	☐	Readings
☐	Group Work	☐	Written Assignment/Essay(s)
X	Student Project	☐	Other:

Evaluation/ Grading

☐	Exam(s)	☐	Class Participation/Discussion Boards
☐	Written Assignment/Essay(s)	X	Class Project(s)
X	Other: Presentation and demonstration of final project in front of an audience		

Topical Outline

Rather than follow a traditional lecture-style/textbook form of teaching, this course is driven by the completion of practical project that builds skills in software engineering and garners experience in implementation and publishing of a real-world application.

1. **Java 2D Graphics API**
    - draw lines, rectangles and other geometric shapes
    - fill shapes with solid colors or gradients and textures
    - draw images, optionally applying filtering operations
    - animate 2D objects independently, and as controlled by a user via the mouse and keyboard
    - detect collisions between 2D objects

2. **Game Design**
    - Addict your user: game mechanics, strategy and playability
    - Telling a story: worlds, characters and levels
    - Can your grandma use it: interaction design
    - Don’t imagine it, try it: iterative development

3. **User eXperience / Usability Testing**
    - how to perform user observation under controlled conditions to determine how well people can use the product
    - prioritizing user feedback to generate development tasks
    - evolutionary product development

4. **Basic Software Engineering**
    - Clean as you go
    - DRY (don’t repeat yourself)
    - Refactoring opportunities and tools
    - Writing clean and readable code