CompuScholar, Inc.

Alignment to Ohio's "145090 Game Design" Standards

Course Title: **TeenCoder: Game Programming**

Course ISBN: **978-0-9887070-1-6**

Course Year: **2015**

Grades: **9th - 12th grade** (high school)

Career Field	Information Technology
Course Name	145090 - Game Design
Description	This course will prepare students to design and program games using commercial and open source programs and applications. Students will learn industry standard programming language constructs to write programs that integrate classes, class methods, and class instances. Students will learn input method handling, animation, collision detection, game physics and basic artificial intelligence.

Note 1: **TeenCoder: Game Programming** is a 2nd-semester course, intended to be paired with the **TeenCoder: Windows Programming** title as a 1st-semester prerequisite. Requirements that are met in the 1st-semetser are marked with "**see TCWP**".

Note 2: The Web-design topics listed in this standard are not generally relevant to Game Design. Those topics are covered in our **KidCoder: Web Design** course and marked as "**see KCWD**".

Strand	2. IT Fundamentals	
Description	Learners apply fundamental principles of IT, including the history of IT and its impact on society, common industry terms, systems theory, information storage and retrieval, database management, and computer hardware, software, and peripheral device configuration and installation. This base of knowledge and skills may be applied across the career field.	
Outcome	**2.4. Emerging Technologies: Identify trending technologies, their fundamental architecture, and their value in the marketplace.**	**CITATION(S)**
2.4.1. Investigate the scope and the impact of mobile computing environments on society.		see KCWD
2.4.2. Describe the differences, advantages, and limitations of cloud computing (e.g., public cloud, private cloud, hybrid cloud) and on-premises computing.		see KCWD
2.4.3. Utilize cloud computing applications (e.g. services, applications, virtual environments).		Students use our cloud-based LMS throughout the course.

Outcome	2.7. Web Architecture: Explain the fundamentals of delivering information and applications using web architecture.	CITATION(S)
2.7.1. Describe methods of securely transmitting data.		see KCWD
2.7.2. Describe ways to present data (e.g., mobile applications, desktop applications, web applications).		see KCWD
2.7.3. Differentiate between a client and a server.		see KCWD
2.7.4. Identify how the use of different browsers and devices affects the look of a webpage.		see KCWD
2.7.5. Explain the relationship between data transmission volumes, bandwidth, and latency.		see KCWD
2.7.6. Describe the characteristics and use of browser plug-ins.		see KCWD
2.7.7. Compare the advantages and disadvantages of running an in-house server or using a service provider.		see KCWD
2.7.8. Describe the difference between static and dynamic sites and the reasons for using each.		see KCWD
Outcome	**2.9. Project Concept Proposal: Develop a project concept proposal.**	**CITATION(S)**
2.9.1. Identify and incorporate branding strategies.		Chapter 2, Lesson 1
2.9.2. Determine the scope and purpose of the project.		Chapter 2, Lesson 1
2.9.3. Determine the target audience, client needs, expected outcomes, objectives, and budget.		Chapter 2, Lesson 1
2.9.4. Develop a conceptual model and design brief for the project.		Chapter 2, Lesson 1
2.9.5. Develop a timeline, communication plan, task breakdown, costs (e.g., equipment, labor), deliverables, and responsibilities for completion.		Supplemental Lesson 1
2.9.6. Develop and present a comprehensive proposal to stakeholders.		Supplemental Lesson 1
Outcome	**2.11. Troubleshooting: Select and apply troubleshooting methodologies for problem solving.**	**CITATION(S)**
2.11.1. Identify the problem.		see TCWP
2.11.2. Select troubleshooting methodology (e.g., top down, bottom up, follow the path, spot the differences).		see TCWP
2.11.3. Investigate symptoms based on the selected methodology.		see TCWP

2.11.4. Gather and analyze data about the problem.		see TCWP
2.11.5. Design a solution.		see TCWP
2.11.6. Test a solution.		see TCWP
2.11.7. Implement a solution.		see TCWP
2.11.8. Document the problem and the verified solution.		see TCWP
Outcome	**2.12. Performance Tests and Acceptance Plans: Develop performance tests and acceptance plans.**	**CITATION(S)**
2.12.1. Create a written procedure agreed by the stakeholders and project team for determining the acceptability of the project deliverables.		see TCWP
2.12.2. Develop a test system that accurately mimics external interfaces.		see TCWP
2.12.3. Develop test cases that are realistic, compare with expected performance, and include targeted platforms and device types.		see TCWP
2.12.4. Develop, perform, and document usability and testing integration.		see TCWP
2.12.5. Make corrections indicated by test results.		see TCWP
2.12.6. Seek stakeholder acceptance upon successful completion of the test plan.		see TCWP
Outcome	**2.13. Rollout and Handoff: Plan rollout and facilitate handoff to customer.**	**CITATION(S)**
2.13.1. Include overall project goals and timelines in the rollout plan.		Supplemental Lesson 1
2.13.2. Communicate rollout plans to key stakeholders in a timely manner.		see TCWP
2.13.3. Conduct final review and approvals according to company standards.		see TCWP
2.13.4. Identify support staff, training needs, and contingency plans in the rollout plan.		see TCWP
2.13.5. Test delivered application to assure that it is fully functional for the customer or user and meets all requirements.		see TCWP
2.13.6. Deliver support and training materials.		see TCWP
Outcome	**5.1. Programming Concepts: Describe programming concepts.**	**CITATION(S)**
5.1.1. Describe how computer programs and scripts can be used to solve problems (e.g., desktop, mobile, enterprise).		see TCWP
5.1.2. Explain how algorithms and data structures are used in information processing.		see TCWP

5.1.3. Model the solution using both graphic tools (e.g., flowcharts) and pseudocode techniques.	see TCWP
5.1.4. Describe, compare, and contrast the basics of procedural, structured, object-oriented (OO), and event-driven programming.	see TCWP
5.1.5. Describe the concepts of data management through programming languages.	see TCWP
5.1.6. Analyze the strengths and weaknesses of different languages for solving a specific problem.	see TCWP
5.1.7. Compare and contrast the functions and operations of compilers and interpreters.	see TCWP
5.1.8. Describe version control and the relevance of documentation.	see TCWP
Outcome **5.2. Computational and String Operations: Develop code that performs computational and string operations.**	**CITATION(S)**
5.2.1. Compare and contrast primitive types of numeric and nonnumeric data (e.g., integers, floats, Boolean, strings).	see TCWP
5.2.2. Identify the scope of data (e.g., global versus local, variables, constants, arrays).	see TCWP
5.2.3. Write code that uses arithmetic operations.	see TCWP
5.2.4. Write code that uses subtotals and final totals.	see TCWP
5.2.5. Write code that applies string operations (e.g., concatenation, pattern matching, substring).	see TCWP
Outcome **5.3. Logical Operations and Control Structures: Develop code that uses logical operations and control structures.**	**CITATION(S)**
5.3.1. Explain Boolean logic.	see TCWP
5.3.2. Solve a truth table.	n/a
5.3.3. Write code that uses logical operators (e.g., and, or, not).	see TCWP
5.3.4. Write code that uses relational operators and compound conditions.	see TCWP
5.3.5. Write code that uses conditional control structures (e.g. if, if-then-else).	see TCWP
5.3.6. Write code that uses repetition control structures (e.g., while, for).	see TCWP
5.3.7. Write code that uses selection control structures (e.g., case, switch).	n/a
5.3.8. Write code that uses nested structures and recursion.	see TCWP

5.3.9. Write code that creates and calls functions.	see TCWP
5.3.10. Code error-handling techniques.	see TCWP
5.3.11. Write code to access data repositories.	see TCWP
5.3.12. Write code to create classes, objects, and methods.	OOP used throughout the course (e.g. Sprite object)

Outcome	**5.4. Integrated Development Environment: Build and test a program using an integrated development environment (IDE).**	**CITATION(S)**
5.4.1. Configure options, preferences, and tools.		Chapter 1, Lesson 4 Chapter 1 Activity
5.4.2. Write and edit code in the IDE.		All chapters
5.4.3. Compile or interpret a working program.		All chapters
5.4.4. Define test cases.		see TCWP
5.4.5. Test the program using defined test cases.		see TCWP
5.4.6. Correct syntax and runtime errors.		Throughout the course, as needed
5.4.7. Debug logic errors.		Throughout the course, as needed

Outcome	**5.5. Programming Conventions: Develop programs using applications security best practices according to information security policies (e.g., cross- site scripting, Structured Query Language [SQL] injection attack, bounds- checking).**	**CITATION(S)**
5.5.1. Develop programs using data validation techniques.		see TCWP
5.5.2. Develop programs that use reuse libraries.		The XNA SDK is used throughout the course
5.5.3. Develop programs using operating system calls.		The XNA SDK is used throughout the course
5.5.4. Develop programs that call other programs.		n/a
5.5.5. Use appropriate naming conventions and apply comments.		see TCWP
5.5.6. Format output (e.g., desktop, mobile, enterprise, reports, data files).		see TCWP

Outcome	5.6. Software Development Lifecycle: Apply the software development lifecycle (SDLC).	CITATION(S)
5.6.1. Determine requirements specification documentation.		see TCWP
5.6.2. Identify constraints and system processing requirements.		see TCWP
5.6.3. Develop and adhere to timelines.		see TCWP
5.6.4. Identify a programming language, framework, and an integrated development environment (IDE).		Chapter 1, Lesson 4 see TCWP
5.6.5. Identify input and output (I/O) requirements.		see TCWP
5.6.6. Design system inputs, outputs, and processes.		see TCWP
5.6.7. Document a design using the appropriate tools (e.g., program flowchart, dataflow diagrams, Unified Modeling Language [UML]).		see TCWP
5.6.8. Create documentation (e.g., implementation plan, contingency plan, data dictionary, user help).		see TCWP
5.6.9. Review the design (e.g., peer walkthrough).		see TCWP
5.6.10. Present system design to stakeholders.		see TCWP
5.6.11. Develop the application.		see TCWP
5.6.12. Compare and contrast software methodologies (e.g., agile, waterfall).		see TCWP
5.6.13. Perform code reviews (e.g., peer walkthrough, static analysis).		see TCWP
5.6.14. Ensure code quality by testing and debugging the application (e.g., system testing, user acceptance testing).		see TCWP
5.6.15. Train stakeholders.		see TCWP
5.6.16. Deploy the application.		Chapter 13, Lesson 4
5.6.17. Collect application feedback and maintain the application.		see TCWP
Outcome	**5.7. Configuration Management: Describe configuration management activities.**	**CITATION(S)**
5.7.1. Explain version management and interface control.		see TCWP
5.7.2. Explain baseline and software lifecycle phases.		see TCWP
5.7.3. Analyze the impact of changes.		see TCWP

Strand	6. Web Development	
Description	Learners apply principles of design and technology, including programming standards and protocols, to create, test, host, and maintain webpages and websites with text, graphics, multimedia, scripting, linking, and data integration in a structure that is easy to navigate and accessible for all users via a variety of hardware and software platforms.	
Outcome	**6.2. Links and Multimedia: Add links to a webpage and insert multimedia files.**	**CITATION(S)**
6.2.1. Create absolute links and relative links.		see KCWD
6.2.2. Write a Hypertext Markup Language (HTML) anchor that links to another section of the same webpage.		see KCWD
6.2.3. Create hyperlinks that send e-mail messages and download files.		see KCWD
6.2.4. Insert image and wrap text around the image using Cascading Style Sheets (CSS).		see KCWD
6.2.5. Resize a graphic image in a webpage using CSS.		see KCWD
6.2.6. Insert audio and video files into a webpage using HTML tags.		see KCWD
6.2.7. Build a hover or mouseover effect to change the style of a link.		see KCWD
Outcome	**6.3. Scripting: Integrate scripting into a webpage.**	**CITATION(S)**
6.3.1. Select and apply scripting languages used in web development.		see KCWD
6.3.2. Insert client-side script into a webpage.		see KCWD
6.3.3. Insert comments into client-side scripts.		see KCWD

Strand	7. Digital Media	
Description	Learners apply principles of digital media to produce interactive media; develop and produce multimedia applications; integrate typography into media; create 3D models and 2D and 3D animation; and create digital video, audio, and photographs.	
Outcome	7.2. Multimedia Tools: Develop navigational structures, scripts, storyboards, and flowcharts for multimedia applications.	**CITATION(S)**
7.2.1. Choose a navigational menu structure (e.g., rollovers, drop-downs, disjointed).		see KCWD
7.2.2. Construct and place navigational units.		see KCWD
7.2.3. Build in interactive elements.		Interactive games are developed throughout the course
7.2.4. Determine uses and needs for site maps, multimedia scripts, storyboards, and flowcharts.		see KCWD
7.2.5. Make preliminary sketches showing placement of images and text on screen.		Chapter 2, Lesson 1
7.2.6. Show placement of buttons and navigational graphics.		Chapter 2, Lesson 1
7.2.7. Provide information on color schemes.		Chapter 2, Lesson 1
7.2.8. Describe music, video, and special effects to be used.		Chapter 2, Lesson 1
7.2.9. Provide a sample layout to stakeholders for review.		Chapter 2, Lesson 1
7.2.10. Select and create visual design elements appropriate for the intended audience and use.		Chapter 2, Lesson 1
7.2.11. Develop characters and narrative to support intended outcomes.		Chapter 2, Lesson 1