School District of Marshfield Course Syllabus

Course Name: Game Programming H
Length of Course: Semester
Credit: 1/2 Credit

Program Goal:

Empower learners to be college and career ready through standards-based experiences in the classroom and career-based learning experiences with business and industry partners. Design and implement educational experiences for creating a skilled, knowledgeable, and productive workforce. Learners will engage in competencies that enable them to stay up-to-date with evolving skills as they pursue careers directly out of high school, as technical school degree earners, or as university graduates. Our goal is to develop critical thinkers and collaborative problem solvers, providing connections to the issues and challenges facing our local, regional, and global economies.

Course Description:

What does it take to be a game developer? This course provides students with an understanding of the principles and concepts of video game development, animation, and app development processes. Students will learn game design theory, animation techniques, and app development processes using state-of-the-art integrated development environments. Students design and develop games, analyze popular games, and learn about various aspects of the game industry. This is a project-based course providing students with several hands-on experiences, providing insight as to what it takes to be a game programmer in today's world.

Standards: Wisconsin Standards for Computer Science (CS)	
Algorithms and Programming	
AP1: Students will recognize and define computational problems using algorithms and programming.	
Develop algorithms. AP1.a	1.a.8.h: Analyze a problem, and then design and implement an algorithmic solution using sequence, selection and iteration. 1.a.11.h: (+) Decompose a large-scale computational problem by identifying generalizable patterns and applying them in a solution.
AP2: Students will create computational artifacts using algorithms and programming.	
Develop and implement an artifact. AP2.a	2.a.12.h: Design, develop, and implement a computing artifact that responds to an event (e.g., robot that responds to a sensor, mobile app that responds to a text message, sprite that responds to a broadcast).
AP4: Students will develop and use abstractions.	
Create and use abstractions (representations) to solve complex computational problems. AP4.a	4.a.12.h: (+) Identify programming language features that can be used to define or specify an abstraction. 4.a.13.h: (+) Identify abstractions used in a solution (program or software artifact) and reuse those abstractions to solve a different problem.
AP5: Students will collaborate with diverse teams.	
Work together to solve computational problems using a variety of resources. AP5.a	5.a.9.h: (+) Use version control systems, Integrated Development Environments (IDEs), and collaboration tools and practices (code documentation) in a group software project.
Impacts of Computing	
IC1: Students will understand the impact and effect computing technology has on our everyday lives.	
Understand the impact technology has on our everyday lives, and the effects of computing on the economy and culture. IC1.a	1.a.6.h: Debate the social and economic implications associated with ethical and unethical computing practices (e.g., intellectual property rights, hacktivism, software piracy, new computers shipped with malware). 1.a.9.h: Describe how computation shares features with art and music by translating human intention into an artifact.
IC2: Students will experience learning within a collaborative, inclusive computing culture and explain the steps needed to ensure that all people have access to computing.	
Test and refine digital artifacts for accessibility. IC2.b	2.b.3.h: Design a user interface (e.g., web pages, mobile applications, animations) to be more inclusive, accessible, minimizing the impact of the designer's inherent bias.

Key Vocabulary:			
abstract	accessibility	animatic	Asymmetric
avatar	beta	blind testing	commercial viability
copyright	creativity	cut scene	development
design	game design	gameplay	graphics
immersion	incremental	intellectual property	iterative
levels	level design	action game	noob
patent	play balance	pitch	prototype
real-time	sandbagging	sandbox	simulation
solitaire	symmetric	theme	trademark
transparency	user interface	VR Virtual Reality	modeling
polygon count	triangle	quad	real-time render
optimization	silhouette	bread crumb	texturing
bump map	normal map	alpha map	light map
decals	shaders	rigging	skinning
one-off animations	looping animations	game engine	game loop
gamestate	player input	waypoints	scripting
2d graphics	2.5D graphics	3D graphics	abandonware
MDA mechanics, dynamics, aesthetics	AI Artificial Intelligence	asynchronous gameplay	AR augmented reality
clipping	developer	emulator	frame rate
game mechanics	multiplier	patch	ping
platform	procedural generation	class	actor
pawns	characters	brush	worlds
bit	mod	lighting	cameras
background	instance	static object	animated object
condition event	action event	code	compile
LOD models (level of detail)	DevOps		

Topics/Content Outline- Units and Themes:

Quarter 1:

- History of video and computer game development (1-2 weeks)
    - 2D and 3D games
- Game Development (5-8 weeks)
    - Narrative Construction
    - Game and Aesthetic Design
    - Programming

Quarter 2:

- Game Engines (3-5 weeks)
- Game Development Projects (6-8 weeks)
- The Gaming Industry (1-2 weeks)
    - Visual Art Roles
    - Programming Roles

Primary Resource(s):	
Android Boot Camp 3rd Edition Cengage Learning ©2016	Microsoft Visual Basic Windows Web Store & Database Apps 1st Edition Cengage Learning ©2018