Questofspook Game Development

Generated from DOCX

2025-10-19 19:12:07 +0000

Quest Of Spook – 2D Game Development

# The Quest of Spook

## A1. Requirement Brief

**Course Details:** **Grade Level:** 9–12

**District:** Computer Science Software Engineering

College Articulation: Articulated College Credit to Mira Costa CC CS 111

**Content Area:** Computer Science / Career and Technical Education (Information Technology Pathway)

**Course Standards:** **National:** CTE (CSTA/IT Pathway)

ISTE Standards for Students - International Society for Technology in Education (2016)

## A2. Learning Goal

Students will collaboratively design and develop a 2D adventure game using JavaScript and the Open Coding Society’s custom game engine. Students will demonstrate understanding of front-end development, object-oriented programming, JSON data structures, and interactive game mechanics while applying digital citizenship and computational thinking principles to create an engaging spooky mansion exploration experience.

### A2a. Elaboration on Goals

In the Quest of Spook project, students transform from code consumers to game creators, building an immersive 2D adventure game that combines technical programming skills with creative storytelling. Using the Open Coding Society’s pages custom game engine located at /assets/js/adventureGame, students will craft a spooky mansion exploration experience that demonstrates mastery of JavaScript fundamentals, JSON object manipulation, and interactive user interface design.

This project emphasizes both individual technical growth and collaborative game development. Students work in teams to design game mechanics, create custom sprites and backgrounds, implement player interactions, and build progressive challenge systems. Through this hands-on approach, they engage with authentic software development practices while creating an entertaining and functional game experience.

### A2b. Prerequisites

Students are familiar with GitHub Pages, HTML, CSS, and basic JavaScript syntax. They have experience with the Software Development Life Cycle (SDLC) including code commits, GitHub pushes, pull requests, and collaborative development practices using the Open Coding Society repository structure.

### A2c. Background

Prior to this project, students have completed foundational lessons in web development and JavaScript programming. They are now ready to apply these skills in a comprehensive game development context, learning advanced JavaScript concepts through practical implementation. The Quest of Spook project serves as a first trimester capstone experience that integrates multiple programming concepts into a cohesive, interactive application.

### A2d. Learning Phases

The project follows a structured six-level progression that mirrors professional game development workflows. Students begin with basic movement and interaction mechanics, advance through scene management and challenge implementation, and culminate in a fully functional adventure game. Each level is built by distinct teams, but the intention is to build consistent game experiences and a purposeful quest. This coordination across teams, to meet a final goal, helps individuals lift skills as they collectively discuss and introduce new concepts in game logic, animation, and user experience design.

## A3. Standards Alignment

### A3a. CTE National Standards (CSTA)

* **CSTA 3A-AP-13:** Decompose problems and subproblems into parts to facilitate design, implementation, and review of programs
* **CSTA 3A-AP-17:** Systematically design and develop programs for broad audiences by incorporating feedback from users
* **CSTA 3A-AP-22:** Design and develop computational artifacts working in team roles using collaborative tools

### A3b. Digital Citizenship Principles (ISTE-aligned)

* **Empowered Learner:** Students design their learning pathways through game customization choices
* **Digital Citizen:** Ethical use of sprites, assets, and collaborative development tools
* **Knowledge Constructor:** Creative problem-solving through game mechanic implementation
* **Computational Thinker:** Efficient game logic design and debugging practices
* **Innovative Designer:** Iterative game design, playtesting, and user feedback integration

## A4. Technology Tools

Students will develop advanced JavaScript programming skills and gain experience with game development frameworks while building upon their existing web development foundation.

### A4a. Game Development Progression

#### Quest of Spook: Six-Level Adventure Framework

Theme: Explore a haunted mansion, solve challenges, and illuminate the darkness — ‘Light your way from curious explorer to mansion master.’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Level** | **Skill Focus** | **Technology Resource** | **Learning Activity** | **Assessment** | **Digital Citizenship Principle** |
| **1. Explore** | JavaScript movement, NPC interaction | VS Code, JavaScript event listeners, OCS Game Engine | Implement player movement controls and dialogue trigger systems in mansion environment | Functional character movement with NPC interaction demo | **Digital Citizen**: Respectful character interaction design and inclusive dialogue creation |
| **2. Choose Room** | Scene management, JSON objects | JavaScript modules, JSON configuration files, Canvas API | Create room transition system using JSON to define room properties and load game scenes | Working room selection with visual scene transitions | **Knowledge Constructor**: Structured data organization and modular programming practices |
| **3. Play Challenge** | Game logic, conditional statements | JavaScript logic operators, OCS challenge framework, debugging tools | Design and implement mini-game challenges using conditional logic and user input validation | Completed challenge with win/lose conditions and feedback | **Computational Thinker**: Logical problem decomposition and algorithm implementation |
| **4. Collect Gem** | Animation, sprite manipulation | CSS animations, JavaScript sprite classes, Canvas rendering | Create gem collection mechanics with animated feedback and torch lighting progression | Functional gem collection with visual animation sequence | **Innovative Designer**: Creative visual feedback and user experience design |
| **5. Return to Mansion** | Data persistence, state management | localStorage API, JSON state serialization, progress tracking | Implement game progress saving and NPC dialogue updates based on player achievements | Persistent game state with updated mansion environment | **Empowered Learner**: Data management and user progress tracking systems |
| **6. Complete Torch** | Final integration, polished gameplay | Git version control, GitHub Pages deployment, user testing | Integrate all systems into cohesive final game with complete mansion illumination sequence | Fully functional game deployed and ready for community showcase | **Innovative Designer**: Complete project integration and public presentation preparation |

### A4b. Technical Implementation Details

**Core Game Engine Components:**

The Open Coding Society game engine provides several foundational classes that students will extend and customize for their Quest of Spook implementation. The **Player Class** handles character movement, collision detection, and interaction states, allowing students to implement smooth navigation through the mansion environment. The **Scene Manager** coordinates room transitions, asset loading, and game state management, ensuring seamless movement between different areas of the mansion.

Students will also work with the **Challenge System**, which provides a mini-game framework with scoring and completion tracking capabilities. The **Animation Engine** manages sprite animations, lighting effects, and visual feedback, particularly important for the torch lighting progression that serves as the game’s central mechanic. Finally, the **Data Manager** handles JSON configuration, progress saving, and asset organization, allowing teams to maintain consistent game data across all levels.

**Customization Elements:**

Students have extensive creative freedom in developing visual and interactive elements for their assigned levels. **Sprites** include custom character designs, non-player characters (NPCs), and interactive objects that populate each room of the mansion. **Backgrounds** encompass room environments, atmospheric effects, and visual themes that establish the spooky mansion aesthetic.

**Audio** components include sound effects, background music, and interaction feedback that enhance the immersive experience. **Challenges** represent the puzzle designs, difficulty progression, and reward systems that make each room unique and engaging for players exploring the mansion.

### A4c. Assessments

**Prototype Development Framework Leading to Night at the Museum:**

* **Checkpoint 1: Storyboard Review** - Teams present visual storyboards showing mansion layout, room designs, character concepts, and overall game flow before beginning development
* **Checkpoint 2: Individual Level Development** - Each team develops and demonstrates their assigned level with core mechanics functional (movement, interactions, challenges, or animations as appropriate)
* **Checkpoint 3: Integration & Mansion Lighting** - Teams integrate individual levels into cohesive mansion experience with working torch lighting progression and scene transitions
* **Checkpoint 4: Night at the Museum Prototype** - Live gameplay demonstration for community audience with functional prototype ready for visitor interaction and feedback collection
* **Final Reflection:** Post-showcase documentation of development process, community feedback analysis, and lessons learned for future development phases

### A4d. Technology Resources

Each level’s technology resources directly support the main learning goal of “collaboratively designing and developing a 2D adventure game using JavaScript and the Open Coding Society’s custom game engine.”

**Learning Goal Component → Technology Resource Alignment:**

* **JavaScript Programming:** VS Code, Chrome DevTools, JavaScript ES6 features enable students to implement game logic, player controls, and interactive mechanics
* **Game Engine Integration:** Open Coding Society adventureGame framework provides structured foundation for sprite management, scene transitions, and collision detection
* **JSON Data Structures:** JSON object creation and manipulation tools allow students to define game assets, level configurations, and player progress data
* **Front-End Development:** HTML5 Canvas, CSS animations, and DOM manipulation support game rendering and user interface design
* **Collaborative Development:** GitHub Projects, Issues, and Pull Requests facilitate team-based game development and code review processes
* **Game Asset Creation:** Sprite editors, background design tools, and animation frameworks enable custom visual content creation

## B. Implementation - E-Learning Environment

### Selected Activity: Level 3 - Play Challenge Implementation

**Activity Selected:** Level 3 - Play Challenge where students design and implement mini-game challenges using conditional logic and user input validation.

**E-Learning Implementation Process:**

This activity is implemented in a collaborative online environment where students work with the Open Coding Society game engine framework. Students access lesson materials, code templates, and game assets through the repository structure, enabling both individual creativity and team collaboration.

**Step-by-Step E-Learning Implementation:**

1. **Initiation:** Students access challenge templates through GitHub Issues linked from Canvas assignments
2. **Tutorial Phase:** Students review game engine documentation and challenge framework examples
3. **Development Phase:** Students use VS Code locally or cloud-based environments to implement challenge logic
4. **Testing Phase:** Students use browser DevTools and live testing to debug and refine game mechanics
5. **Collaboration:** Students share challenge designs through GitHub Pull Requests for peer review and feedback
6. **Integration:** Instructor merges approved challenges into main game branch for comprehensive testing

**E-Learning Benefits:** Students gain experience with professional game development workflows, build portfolios of playable games, and develop both technical and creative skills in an engaging, project-based context.

## B1. Digital Tools

### Tool 1: Open Coding Society Game Engine + GitHub Development Workflow

**Purpose:** Game development framework with collaborative version control

**Implementation:**

* **Game Engine Setup:** Students clone repository and access /assets/js/adventureGame framework with pre-built classes for sprites, scenes, and game logic
* **Development Environment:** Local VS Code setup with Live Server extension for real-time game testing, or cloud-based development for Chromebook users
* **Version Control:** Students create feature branches for each game level, use GitHub Issues for task tracking, and submit Pull Requests for code review
* **Asset Management:** Organized folder structure for sprites, backgrounds, audio, and JSON configuration files
* **Collaboration Tools:** GitHub Projects for milestone tracking, code review process for quality assurance, and shared asset libraries

**E-Learning Advantage:** Professional game development workflow, immediate playtesting capabilities, collaborative asset sharing, and portfolio-ready project documentation.

### Tool 2: JavaScript Development Stack + Browser-Based Testing

**Purpose:** Complete front-end game development and testing environment

**Implementation:**

* **Local Development:** Students use JavaScript ES6 features, HTML5 Canvas API, and CSS animations within VS Code environment
* **Browser Testing:** Chrome DevTools for real-time debugging, performance monitoring, and game logic verification
* **Asset Creation:** Integration with sprite editors and background design tools for custom game content
* **Deployment Pipeline:** GitHub Pages automatic deployment for live game sharing and community feedback
* **Performance Optimization:** Browser-based profiling tools for game performance analysis and optimization

**E-Learning Advantage:** Immediate visual feedback through browser testing, accessible development tools, live deployment for sharing, and authentic web development experience that prepares students for industry practices.

## C. References

[CSTA, 2017] Computer Science Teachers Association. (2017). CSTA K–12 Computer Science Standards. Retrieved from <https://www.csteachers.org/page/standards>

[ISTE, 2016] International Society for Technology in Education (ISTE). (2016). ISTE Standards for Students. Retrieved from <https://www.iste.org/standards/for-students>

**[OCS, 2025]** Open Coding Society. (2025). *Open Coding Society Game Engine Repository.* Retrieved from <https://github.com/Open-Coding-Society/pages>