Software Requirements Specification (SRS)
Puzzler

Team: Chris Peters, Max Fitzgerald, Shkamb Tafarshiku, Elijah Miles
Authors: Chris Peters, Max Fitzgerald, Shkamb Tafarshiku, Elijah Miles
Customer: James Daly
Instructor: James Daly

1 Introduction

This Software Requirements Specification (SRS) document provides a detailed description of Puzzler, including its purpose, scope, overall system behavior, constraints, and specific functional and non-functional requirements. It also defines the system's context, and major use cases. The document is intended to guide development and validation.

Purpose

The purpose of this SRS is to clearly define the functional and non-functional requirements of Puzzler, an educational puzzle game for middle-school students, so that developers, and customers share a common understanding of the system's behavior. This document serves as the basis of agreement between the development team and the customer.

Intended Audience:

- Customer/Instructor (James Daly)
- Development team members

Scope

Product: *Puzzler*

Puzzler is a software application designed to assist students, grade 7-8, with learning various subjects. It provides interactive puzzles across three core subject areas: math, science, and English, aligned with age-appropriate difficulty levels. The project belongs to the educational technology (EdTech) domain, specifically focusing on game-based learning.

Primary Goals:

- Reinforce academic skills through fun, game-based learning.
- Encourage problem-solving, critical thinking, and exploration.
- Provide a structured but engaging experience suitable for classroom or independent use.

The Product Will:

- Present puzzles organized by subject and difficulty.
- Track user progress and puzzle completion.
- Provide teachers or users with performance summaries.
- Offer hints, instructions, and encouragement to players.

- Allow users to retry puzzles or advance when successful.

The Product Will Not:

- Serve as a full learning management system.
- Provide graded or official assessments.
- Require advanced subject knowledge beyond middle school curriculum.

Definitions, acronyms, and abbreviations

- **Puzzle:** A unique instance that presents an educational challenge requiring reasoning or subject knowledge.
- **Level:** A segment of the game consisting of a map and its associated puzzles.
- **Map:** The in-game world presented to the player contains game-objects and access to each puzzle.
- **Key:** Rewarded to player after completing a puzzle.
- **Player-character:** The in-game character that the user controls and navigates around the map.
- **UI/UX:** User Interface, the compilation of graphical elements presented to the user within the game.
- **HUD:** [Heads Up Display] Part of the UI, In-game overlay that displays information to the player such as map progress, key progress, prompts, information, hints, and interaction queues.
- **Session:** The period that a user spends playing the game.
- **Hint System:** Game feature that provides partial guidance during puzzles in response to incorrect attempts at completing the puzzle.

Organization

- **Section 2 – Overall Description:** Context, user characteristics, system environment, assumptions
- **Section 3 – Specific Requirements:** Detailed functional and non-functional requirements
- **Section 4 – Modeling Requirements:** Use cases, diagrams, class models, sequences, and scenarios
- **Section 5 – Prototype:** Information on the prototype and example interactions

- Section 6 – References: Supporting documents
- Section 7 – Point of Contact: Contact information

2 Overall Description

This section provides a high-level overview of the design environment, interfaces, user needs, and constraints.

3 Product Perspective

The proposed system is an educational puzzle game developed in Unity. It is a standalone application designed to run on desktop and laptop platforms. The game integrates multiple subject-based challenges, math, and science, appropriate for middle-school learners. Each puzzle is presented within themed levels or “maps,” and players progress linearly or semi-linearly through the content by successfully solving the required challenges. Unity’s engine provides the game framework, scene management, UI system, physics, and input handling needed to deliver an interactive and visually engaging learning experience. The game does not rely on external servers for gameplay.

4 Product Functions

The major functions of the system include:

- Puzzle Interaction: Players complete subject-based puzzles designed around middle-school academic standards.
- Level/Map Progression: Completing puzzles unlocks additional levels through the GameManager system.
- Player State Management: Player progress, puzzle completion, earned items, and unlocked content are saved and restored through the SaveSystem.
- Feedback and Learning Support: The game provides hints, explanations, or corrective feedback when players struggle with specific challenges.
- User Interface Navigation: Players can navigate menus, settings, and help screens.

5 User Characteristics

The primary users are middle-school students (typically ages 11–14). These users:

- Have varying reading and comprehension skills.
- Are familiar with basic computer or mobile device operation.
- Benefit from clear instructions, visual cues, and intuitive controls.
- May have differing proficiency levels across Math, Science, and English.

6 Constraints

- Development is constrained by Unity's built-in systems, asset pipeline, and performance limitations on target platforms.
- All puzzles must align with typical middle-school academic expectations.
- Visual and gameplay design must remain accessible and age appropriate.
- The game must function offline for classroom use unless networking features are intentionally added.

7 Assumptions and Dependencies

- Users will have access to a compatible device capable of running Unity-built applications.
- Users have basic literacy and digital-navigation skills.
- The Unity engine and required packages (UI toolkit, animation tools, input system) will remain stable and supported.
- Subject matter content is based on general middle-school curricula and does not require region-specific customization unless added later.

8 Apportioning of Requirements

The following requirements are considered out of scope for the current release of Puzzler but may be addressed in future versions:

- Teacher dashboard: A history of the data that each student did in the game, including statistics to show how a class of students may have done on the game. The median scores, highest / lowest scores, and the time each game was completed could be involved here.
- User account system: A system that allows all users to log into their own account, where all accounts are tracked and data is stored. The current version assumes a single local player. This system would allow the developers to make changes to each student's individual account if need be. For example, if a student accidentally trashed an item, the developers would be able to go into that student's account to replace it.
- Online / networking features: Any online leaderboard, cloud save, or teacher-student interaction. The current system operates offline.
- Content expansion and customization: Future maps and puzzles are not included in this version. This prototype only includes 2-3 maps, with 2-3 puzzles per map. Future versions could also allow teachers to customize their own levels for their students, perhaps by inputting their own puzzles. The code would then generate a map that uses data from user input.
- Other platforms: The current version targets Windows desktop via Unity. Future versions of the prototype could include ports to mobile, console, or web.

9 Specific Requirements

1. The system shall provide the user with an interactive educational experience (The game)
   A. The game shall provide 2–3 distinct maps (levels)
      i. The game shall include 2–3 puzzles per map.
         a. Each puzzle shall only target one subject
            ii. Each puzzle shall only target one learning objective in that subject
         b. Each puzzle shall contain elements the user will interact with to solve
         c. Upon completing a puzzle,
            i. The user shall be granted an in-game item to indicate puzzle completion (referred to as a 'key')
            ii. Be presented with a puzzle summary
      ii. Subsequent maps shall contain puzzles that are more difficult than previously completed ones. Difficulty can scale according to one or more of the following criteria:
         a. Number of steps required to solve
         b. Complexity of concepts involved
         c. Amount of information provided vs. required
         d. Utilizing concepts that require understanding the learning objective from puzzles presented in previous maps
   B. The game shall facilitate transition between maps
      i. The next map, and all subsequent maps, will initially be unaccessible to the player
      ii. The next map must be 'unlocked' by completing all puzzles on current map, thus earning all puzzle keys
      iii. Upon earning all puzzle keys on current map, The game shall allow the user to transition to the next map.
      iv. Upon completion of all puzzles on the final map, The game shall allow the user to end the game.
   C. The game shall provide a player character with the following features:
      i. Capable of movement to traverse the area within the bounds of the map
      ii. Capable of interactions with other game objects found on the map
   D. The game shall provide interactable game objects that perform the following functions:
      i. Act as a barrier that the player character cannot traverse through (i.e. walls)
      ii. Act as an initiating event for each puzzle that transitions the game to puzzle

screen.

E. The game shall have a Main Menu presented on startup with the following options available:

i. CONTINUE – Allow the user to continue playing the most recent saved game

ii. NEW GAME – Allow the user to start a new play through

iii. LOAD GAME – Allow the user to continue playing from a save file

iv. SETTINGS – Allow the user to adjust certain in-game options

v. QUIT – Allow the user to exit the program

F. The game shall facilitate saving game progress

i. User will be able to manually create a save file

ii. The system shall automatically create a save file upon completing a puzzle

G. The game shall be able to recreate the state of play at the point when a save file was created.

i. Keys unlocked

ii. Maps unlocked

iii. Player location on map

iv. Puzzle summaries from previously completed puzzles

H. The game shall provide a how-to-play/controls summary page upon beginning a new game.

I. The game shall provide a Heads-Up-Display(HUD) overlay that persists during map traversal and puzzle attempts

i. The HUD shall contain areas to present text to the user

a. When a player is within range over interacting with a game object, a prompt will appear informing the player of what keyboard key to use to interact with the game object.

b. During puzzles, hints will be displayed to the user

ii. While attempting puzzles, the HUD shall provide the user access to any materials initially presented(the prompt) or required to complete the puzzle.

iii. The HUD shall contain fields to indicate Game progression:

a. Map progression – display indicators for:

1) Completed maps

2) Current map

3) Remaining maps

b. Key progression – Display indicators for:

1) Placeholders for unearned keys

2) Earned keys

iv. The HUD shall contain interactable icons:

a. To toggle ON/OFF the background music

b. To provide the user a means to request a hint during puzzles

c. To access previously completed puzzles summaries

d. To save game progression

e. To return to main menu

f. To view the how-to/controls summary

2. The puzzles in the game will utilize educational content

A. Puzzle content shall be aligned with 7th–8th grade learning standards in these subjects:

i. Life Science

ii. Mathematics

iii. English

B. Puzzles shall provide the user a text prompt containing information necessary for solving the puzzle

i. This text will be accessible to the user during the entire course of their attempt at the puzzle.

ii. Text will be presented at, or below, an 8th grade reading level

3. The puzzles shall provide feedback to the user upon submitting their attempted solution

A. The game shall track the number of incorrect attempts for each puzzle.

B. The game shall provide tiered hints after incorrect attempts:

i. After 2 incorrect attempts — give a first-tier hint offering general guidance.

ii. After 3 total incorrect attempts — give a second-tier hint with more specific direction.

iii. After 4 total incorrect attempts — give a third-tier hint giving near-complete guidance.

C. Feedback is provided via the HUD indicating attempt was “Incorrect” or “Correct”.

D. The game shall display puzzle summaries after successfully completing a puzzle:

i. Summarize the learning objective addressed

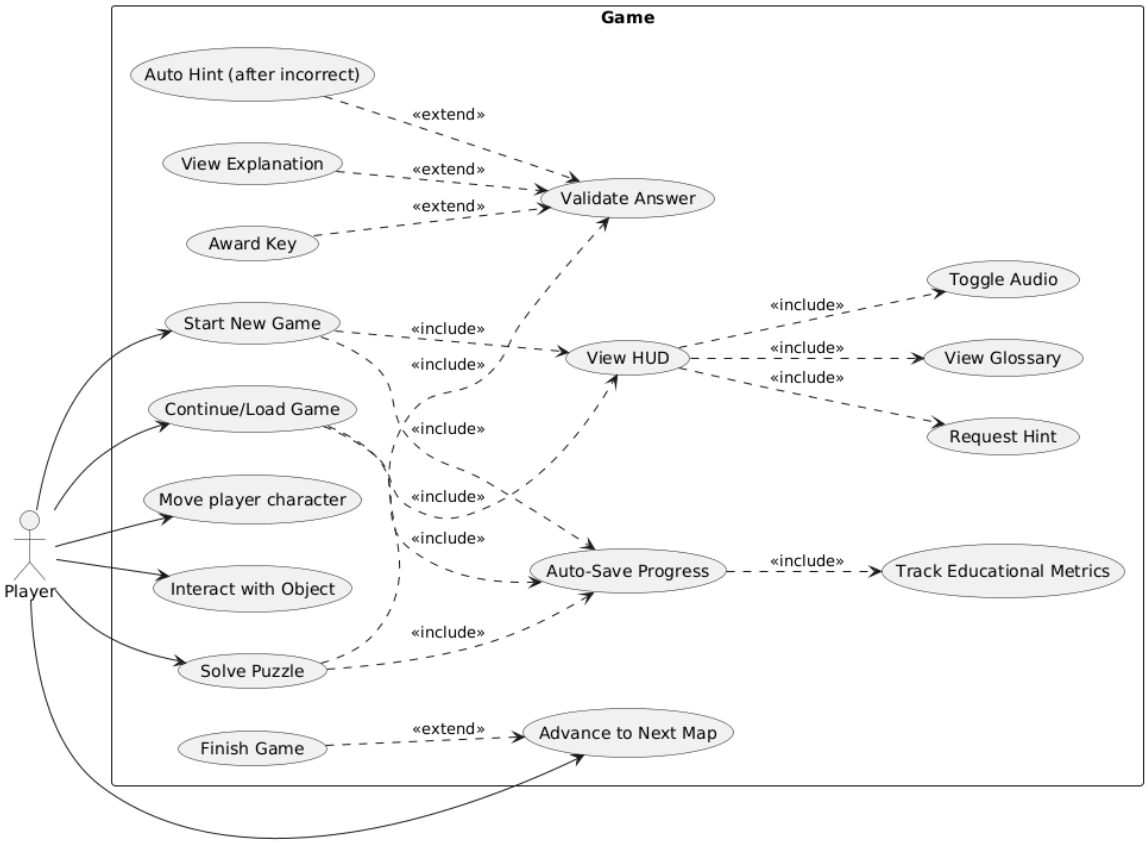
ii. Explain the correct solution method

E. The game shall allow retry attempts without limitation.

F. The game shall prompt the user to “try again” via the HUD after incorrect attempts

10 Modeling Requirements

Use Case Diagram:



Use Case Name:	Start New Game
Actors:	Player
Description:	The player starts a brand-new playthrough from the main menu, creating an initial game state with only the first map unlocked and no puzzles completed or keys collected.
Type:	Primary
Includes:	View HUD, Auto-Save Progress
Extends:	None
Cross-refs:	Requirements for game startup, initial state, map unlocking, and saving

Uses cases:	Precedes Auto-Save Progress, leads to Track Educational Metrics

Use Case Name:	Continue / Load Game
Actors:	Player
Description:	The player resumes a previously saved playthrough from the main menu. The system loads saved progress (unlocked maps, completed puzzles, collected keys, player position, and settings) so gameplay can continue from where it left off.
Type:	Primary
Includes:	View HUD, Auto-Save Progress
Extends:	None
Cross-refs:	Requirements for loading and resuming progress
Uses cases:	Precedes Auto-Save Progress, leads to Track Educational Metrics

Use Case Name:	Solve Puzzle
Actors:	Player
Description:	The player interacts with a puzzle object in the map, opens the puzzle screen, submits one or more answers, optionally requests hints, and ultimately either solves the puzzle or exits. On a correct solution, the player is awarded a key.
Type:	Primary
Includes:	Auto-Save Progress, Validate Answer
Extends:	None
Cross-refs:	Puzzle and feedback requirements
Uses cases:	Precedes Validate Answer, leads to Auto Hint (after incorrect), View Explanation, Award Key

Use Case Name:	View HUD
Actors:	Player
Description:	The player views and uses the in-game head-up display (HUD), which shows current map, key progression, prompts, and access to options.
Type:	Supporting
Includes:	Request Hint, View Glossary, Toggle Audio
Extends:	None
Cross-refs:	UI/usability and settings requirements

Uses cases:	Included by Start New Game, Continue/Load Game; provides access to Request Hint, View Glossary, Toggle Audio

Use Case Name:	Advance to Next Map
Actors:	Player
Description:	After collecting all required keys on the current map, the player unlocks and transitions to the next map. When all maps have been completed, the game ends and a completion state is shown.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Map progression and win-condition requirements
Uses cases:	Leads to Finish Game

Use Case Name:	Move Player Character
Actors:	Player
Description:	The player uses input controls to move the character around the map, navigate to different rooms, and reach interactive objects or exits.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Movement and control requirements
Uses cases:	None

Use Case Name:	Interact with Object
Actors:	Player
Description:	The player interacts with an object in the environment (such as a puzzle portal, chest, sign, or door) to trigger associated behavior like opening a puzzle screen or showing information.
Type:	Primary
Includes:	None
Extends:	None
Cross-refs:	Object interaction and environment requirements

Uses cases:	Follows Move player character; may lead to Solve Puzzle or other context-specific feedback

Use Case Name:	Finish Game
Actors:	Player
Description:	The player reaches the end condition for the current run (for example, obtaining all required keys and reaching the final exit). The system finalizes the current map or game session.
Type:	Primary
Includes:	None
Extends:	Advance to Next Map
Cross-refs:	Game completion and end-of-level requirements
Uses cases:	May follow Solve Puzzle or exploration; extends Advance to Next Map to trigger progression when finish conditions are met

Use Case Name:	Auto Hint (after incorrect)
Actors:	Player
Description:	After the player submits an incorrect answer, the system may automatically present a contextual hint to guide the player toward the correct solution.
Type:	Supporting
Includes:	None
Extends:	Validate Answer
Cross-refs:	Hint system and adaptive feedback requirements
Uses cases:	Triggered as an extension from Validate Answer when an answer is incorrect

Use Case Name:	View Explanation
Actors:	Player
Description:	The player views a step-by-step explanation or solution breakdown for the puzzle, helping them understand why the correct answer is correct.
Type:	Supporting
Includes:	None

Extends:	Validate Answer
Cross-refs:	Educational feedback and explanation requirements
Uses cases:	Triggered as an extension from Validate Answer after checking an answer (usually after completion or on request)

Use Case Name:	Award Key
Actors:	Player
Description:	When the player solves a puzzle correctly, the system grants a key (or equivalent progression token) and updates the player's key count and progression status.
Type:	Supporting
Includes:	None
Extends:	Validate Answer
Cross-refs:	Key progression and unlock requirements
Uses cases:	Triggered as an extension from Validate Answer upon a correct solution; contributes to conditions for Finish Game / Advance to Next Map

Use Case Name:	Validate Answer
Actors:	Player
Description:	The system checks the player's submitted puzzle answer(s) against the correct solution, determines correctness, and records the result for feedback and progression.
Type:	Supporting
Includes:	None
Extends:	None
Cross-refs:	Puzzle validation and scoring requirements
Uses cases:	Included by Solve Puzzle; extended by Auto Hint (after incorrect), View Explanation, and Award Key

Use Case Name:	Toggle Audio
Actors:	Player
Description:	The player turns game audio on/off or adjusts simple audio settings from the HUD, improving comfort and accessibility.

Type:	Supporting
Includes:	None
Extends:	None
Cross-refs:	Audio and accessibility requirements
Uses cases:	Included via View HUD; may be used at any time while HUD options are visible

Use Case Name:	View Glossary
Actors:	Player
Description:	The player opens an in-game glossary or help panel that explains mathematical terms, symbols, and concepts used in the puzzles.
Type:	Supporting
Includes:	None
Extends:	None
Cross-refs:	Glossary and instructional content requirements
Uses cases:	Included via View HUD; supports player understanding of puzzles and explanations

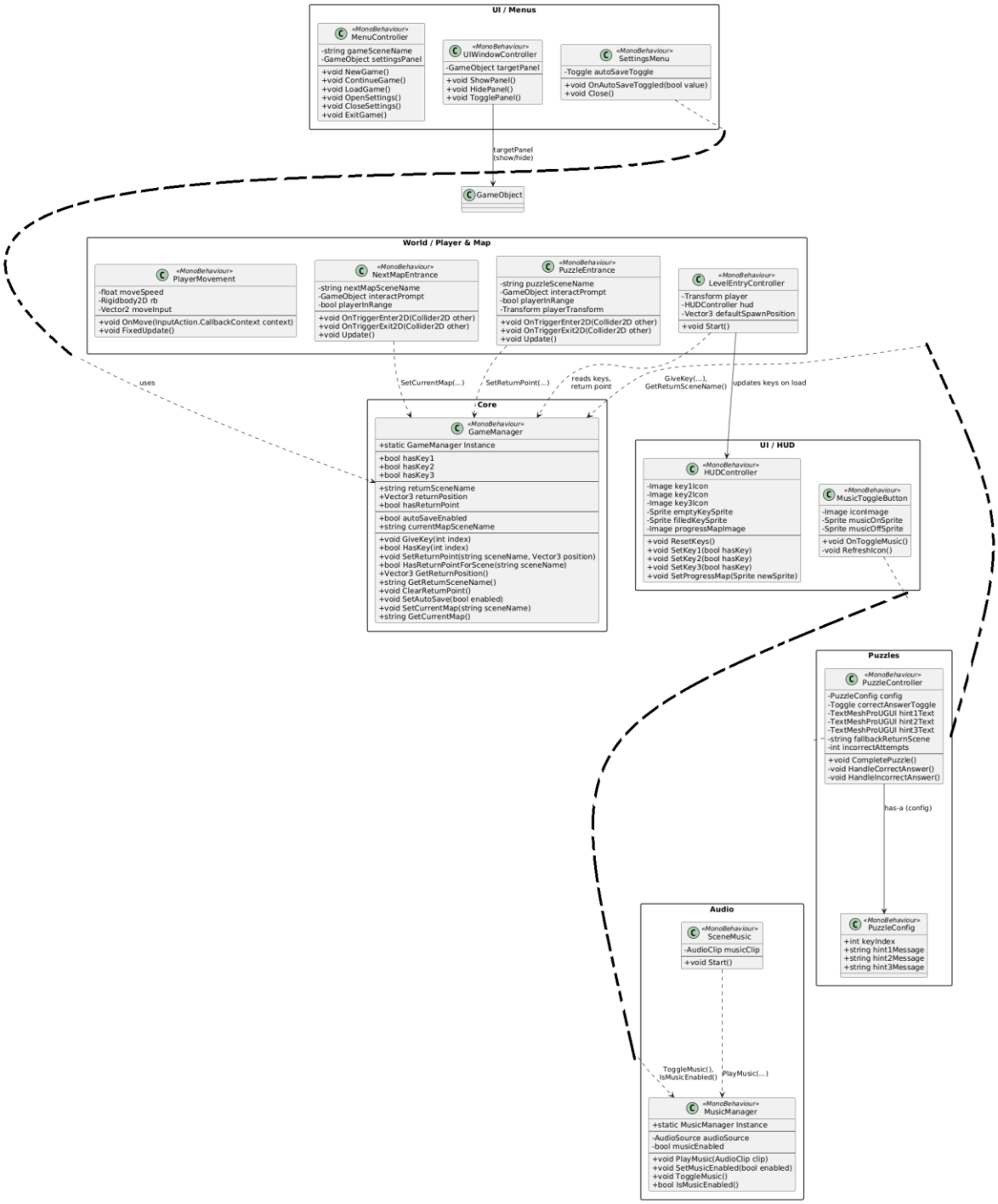
Use Case Name:	Request Hint
Actors:	Player
Description:	The player explicitly requests a hint from the HUD or puzzle screen. The system provides a clue that does not fully reveal the solution but narrows the search space.
Type:	Supporting
Includes:	None
Extends:	None
Cross-refs:	Hint system and assistance requirements
Uses cases:	Included via View HUD (and functionally related to Solve Puzzle); may lead into Auto Hint–style feedback logic

Use Case Name:	Auto-Save Progress
Actors:	Player

Description:	The system periodically saves the player's state (map, position, key count, puzzle completion, and relevant metrics) without requiring manual input, so progress is not lost.
Type:	Supporting
Includes:	Track Educational Metrics
Extends:	None
Cross-refs:	Saving, persistence, and data integrity requirements
Uses cases:	Included by Start New Game, Solve Puzzle, and Continue / Load Game, leads to Track Educational Metrics

Use Case Name:	Track Educational Metrics
Actors:	Player
Description:	The system records educational statistics such as puzzles attempted, accuracy, time to solve, and hint usage for later analysis or instructor reporting.
Type:	Supporting
Includes:	None
Extends:	None
Cross-refs:	Educational metrics and analytics requirements
Uses cases:	Included by Auto-Save Progress; data may be used for external reporting or dashboards

Class Diagram:



UI / Menus

Class Name	MenuController
Attributes	GameSceneName: What map scene to load when user presses ‘New Game’ or ‘Continue’ SettingsPanel: panel who’s visibility is toggled to visible upon pressed the ‘Settings’ button. Toggle to invisible when user pressed the ‘X’ close button.
Operations	Handle functionality related to the Main Menu to enable transitioning between panels and scenes
Relationships	none

Class Name	UIWindowController
Attributes	TargetPanel: panel to have it’s visibility toggled.
Operations	Used in the map scenes to toggle the ‘Controls’ panel
Relationships	none

Class Name	SettingsMenu
Attributes	AutoSaveToggle: user may select to enable/disable auto saves
Operations	Control center for what customizable settings the user has selected.
Relationships	none

World / Player & Map

Class Name	PlayerMovement
Attributes	MoveSpeed: player speed Rb: reference object for unity to control placement and movement of player character
Operations	Used by unity to move player character

Relationships	none

Class Name	NextMapEntrance
Attributes	NextMapSceneName: the name of the map scene to load Interactprompt: The text field on the HUD used to communicate interaction possibilities to the user PlayerInRange: a true/false test if the player is close enough to interact with the game object (box collision test)
Operations	Acts a transition between current map and next map
Relationships	GameManager: manages map attributes such as ‘current map’

Class Name	PuzzleEntrance
Attributes	PuzzleSceneName: the name of the puzzle scene to load Interactprompt: the text field on the HUD used to communicate interaction possibilities to the user PlaerInRange: a true/false test if the player is close enough to interact with the game object (box collision test) PlayerTransform: x,y,z location data for where the player character was on the map prior to transitioning to puzzle scene
Operations	Transitions the game from the map scene to the puzzle scene
Relationships	GameManager: manages map attributes such as ‘returnSceneName’, ‘returnPosition’, and ‘hasReturnPoint’

Class Name	levelEntryController
Attributes	Player: x,y,z location data for where to spawn a player into a new map Hud: communicates with hud DefaultSpawnPosition: if no other player transform provided, a default position is set
Operations	On first spawning in new map, used to setup player positioning
Relationships	GameManager: communicates state variables such as keys earned on this map HUDController: updates HUD with key data

Core

Class Name	GameManager
Attributes	Instance: The gamemanager object being used to control the game (self reference) HasKey1,2,3: Has the player earned the respective key ReturnSceneName: used when transitioning between a map and it's puzzles. ReturnPosition: where to draw the player character upon transition HasReturnPoint: checks for returnPosition otherwise uses default location. AutoSaveEnabled: setting used to control saving after puzzle completion (controlled in settings menu from main menu) CurrentMapSceneName: the name of the current map scene
Operations	Acts as a holder of information that must survive through multiple scene transitions. This data will be used to create save files and to load save files.
Relationships	UI elements to update player accomplishments and game objects that require persistent data through scene transitions

UI / HUD

Class Name	HUDController
Attributes	Key1,2,3Icon: ui element to display empty key placeholder sprite and earned key sprite EmptyKeySprite: used to update key1,2,3Icon, displays that player has not earned that key FilledKeySprite: player has earned this key indicator ProgressMapImage: after completing a map this will be updated to indicate which maps have been completed
Operations	Display relevant progress data to user
Relationships	LevelEntryController: updates scene with user progress data

Class Name	MusicToggleButton

Attributes	IconImage: currently displayed sprite MusicOnSprite: displayed when user enables music MusicOffSprite: displayed when user disables music
Operations	Manages transitions for the music icon on the HUD
Relationships	MusicManager: tells manager if music should be played

Audio

Class Name	SceneMusic
Attributes	MusicClip: the selected audio clip to play on this map
Operations	Used to interface a scene with the music manager
Relationships	MusicManager: provides what music to play to the manager

Class Name	MusicManager
Attributes	Instance: self-reference AudioSource: Unity object that plays the music
Operations	Manages what music is playing and if it should be playing
Relationships	Musictogglebutton: should music be playing SceneMusic: what music is to be played

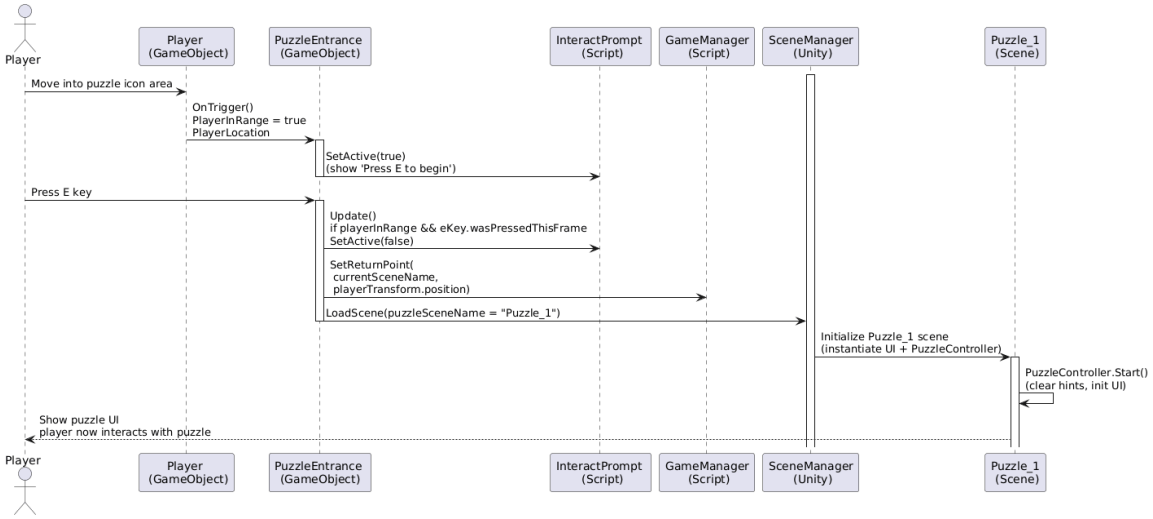
Puzzles

Class Name	Puzzlecontroller
Attributes	Config: script that houses data for the puzzle such as what key will be awarded by this puzzle and what the text for the hints will be. CorrectAnswerToggle: temporary feature Hint1,2,3Text: The text to give the user when requesting hints/being given hints FallbackReturnScene: the scene to return to after leaving puzzle scene IncorrectAttempts: tracking variable
Operations	pass the unique puzzle elements to the puzzle scenes through this generalized interface

Relationships	PuzzleConfig: where the data for the puzzle comes from GameManager: communicates return scene name

Class Name	PuzzleConfig
Attributes	Keyindex: the key to award for completing the puzzle Hint1,2,3Message: text of the hints to provide the user
Operations	Data structure
Relationships	Puzzlecontroller: passes data to this interface

Sequence 1:



Description: Player enters the puzzle.

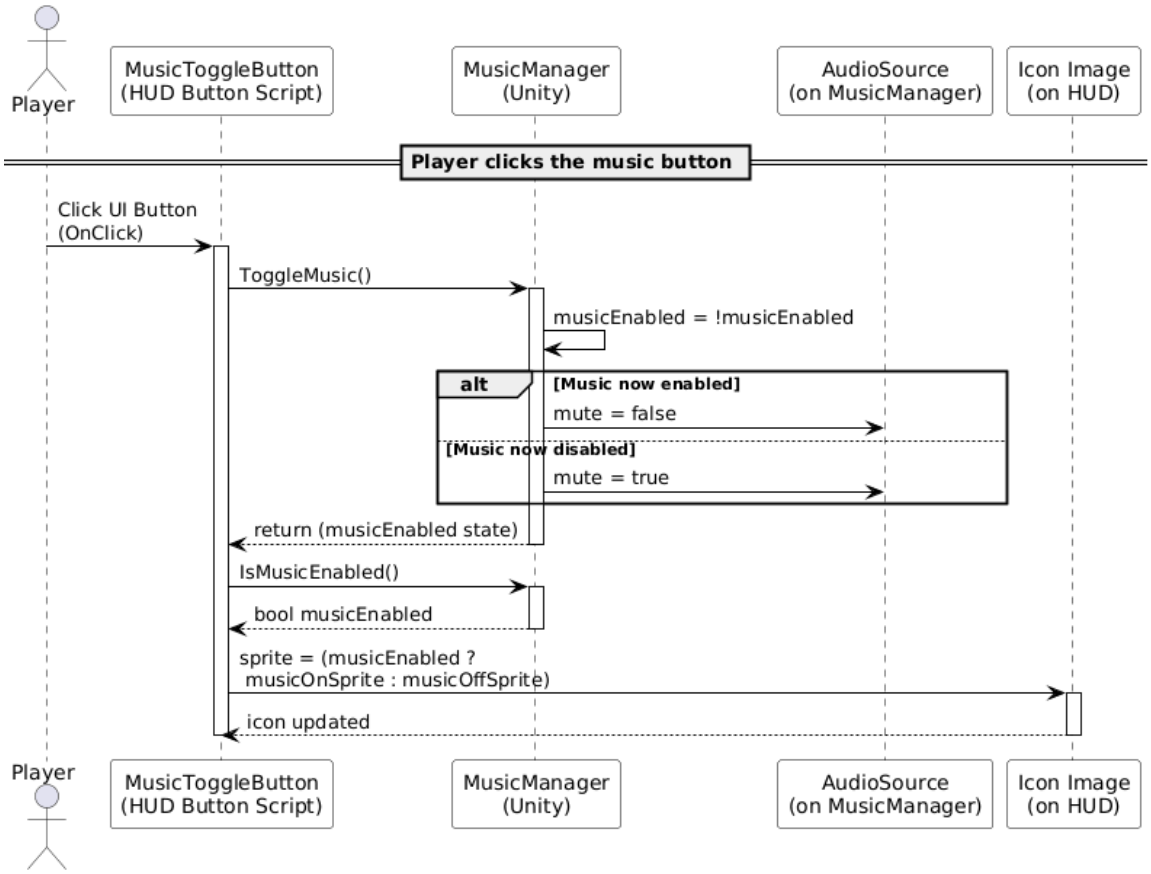
User moves player character onto the puzzle entrance icon. Prompt is displayed on HUD “Press E to enter...” to inform the user of how to proceed.

User pressed E key. Prompt is removed from the HUD and the GameManager saves information about the current map and where the player character is located on it. Puzzle

Entrance contains what scene to load in order to launch the puzzle and this operation is executed via Unity's scene manager.

The user performs the puzzle and then completes it and is returned to the previous map at the same location.

Sequence 2:

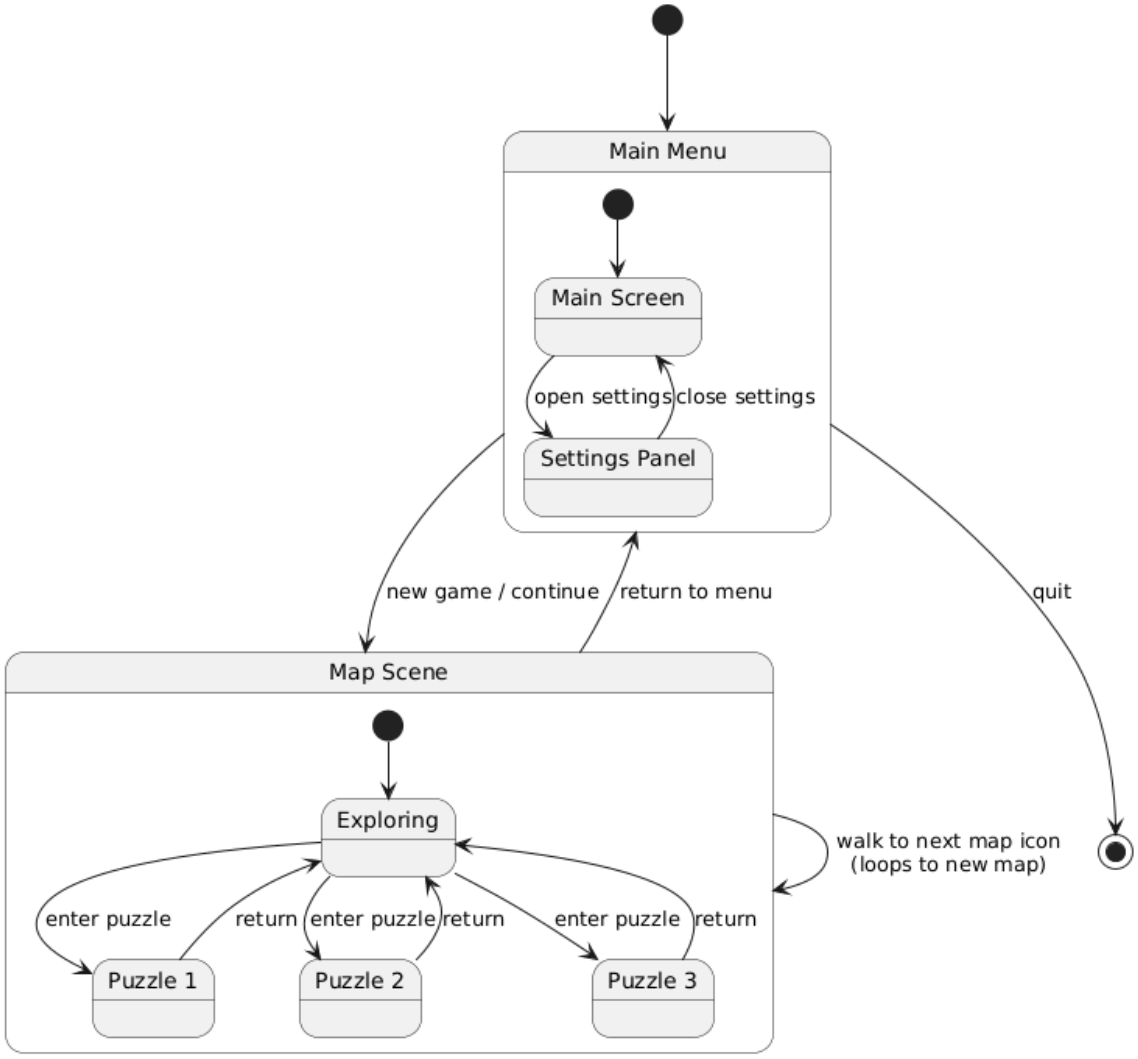


Description: Toggling the Music enable/disable icon

Map is the active scene

User moves the mouse over toggle icon and left clicks. ToggleMusic() is called onClick() and MusicManager handles the toggle of mute state. ToggleButton script checks status of music enabled and alters the display to properly indicate the status of music playing on the map.

State Diagram:



This state diagram represents the flow of an educational game prototype built in Unity. The game starts at the Main Menu, which contains the main screen with game options and a settings panel. From the Main Menu, players enter the first map when starting a new game, or return to whichever map they left off at when continuing.

Each map contains a starting area and three puzzle scenes. Players can navigate from the starting area to any puzzle, and return to the starting area after completing it. When ready, players walk to a "next map icon" to progress to the next map, which follows the same structure. Players can return to the Main Menu from any map at any time, or quit the game from the Main Menu.

Prototype

The prototype begins with a ‘Main Menu’ scene. From here it is possible to select the ‘New Game’, ‘Settings’, and ‘Quit’ options to view functionality. ‘Settings’ will open a new window to display a list of possible game settings that are exposed to the user for alteration. ‘Quit’ will simply end the program. ‘New Game’ will launch a new instance of the game beginning at the ‘Starting_Map’.

Here it is possible to see and navigate the player character on screen to explore the current map. The ‘Starting_Map’ consists of a large grassland area and a single building which the player stands within upon starting a new game. Navigating the player downwards will allow the player to exit the building through a gap in the walls, indicated by the grass tile separating two wall tiles. Just outside the building there are 3 icons on the ground with purple question marks (?). These icons indicate a puzzle entrance, approaching any of these icons will cause a prompt to be displayed on the Heads-Up-Display (covered next) “Press E to begin puzzle...”. Pressing ‘e’ key on the keyboard will launch the puzzle scene linked to that entrance.

The HUD is an overlay covering the top and bottom of the screen surrounding the viewing window (player-character + map). On the HUD there are a number of features. Beginning at the top-left there is a sprite with 3 circles connected by a line. This is to indicate to the player their current ‘Map progress’ and shows how many maps the player has completed. On the top-right there are 3 key slots placeholders; these indicate to the player which keys they have earned through completing puzzles on that map. Moving on to the bottom segment of the HUD there are a number of icons on the left side. These consist of (Left to right): Back, Toggle game music, save, controls, puzzle summaries. Currently the controls and toggle game music are functional. Pressing the controls icon will cause a pop-up window to appear displaying to the player what keys are used to navigate and interact in the world. As well as a glossary for the icons.

Currently, the puzzle scenes consist of a check box to simulate the correct answer and a submit button labeled ‘Complete Puzzle’. While leaving the check box unchecked, press the complete puzzle button. Upon first click there will be no change however for the next 3 consecutive presses, hints shall appear on the lower portion of the HUD. All subsequent presses after 3 hints have appeared will not produce more hints, there are only 3 per puzzle. Checking the check box now and pressing the complete puzzle button will return the player to the previous map. Notice there is also a new icon on the HUD, a circle with an H, this is to manually request a hint(non-functional).

How to Run Prototype

Prerequisites

- Unity Hub installed on your system (download from unity.com if needed)
- Unity Version 6000.2.10f1 installed through Unity Hub

Installation Steps

- Navigate to the project's GitHub repository at Project Repo.
- Clone the repository using command: `git clone https://github.com/DeepOceanCobaltBlue/SWE_final.git` or alternatively download the project as a ZIP file using the "Code" button on GitHub and extract it to a local directory.
- The project folder structure should contain an Assets folder and Packages folder at the root level

Opening in Unity

- Launch Unity Hub.
- Open Unity Hub and click "Add" or "Open" to select a project from disk
- Navigate to and select the root project folder (SWE_final) that contains the Assets folder.
- Unity Hub will display the project with Unity version 6000.2.10f1
- Click on the project to open it in Unity Editor
- Initial project import may take 5-10 minutes as Unity compiles and imports assets.

Running the prototype

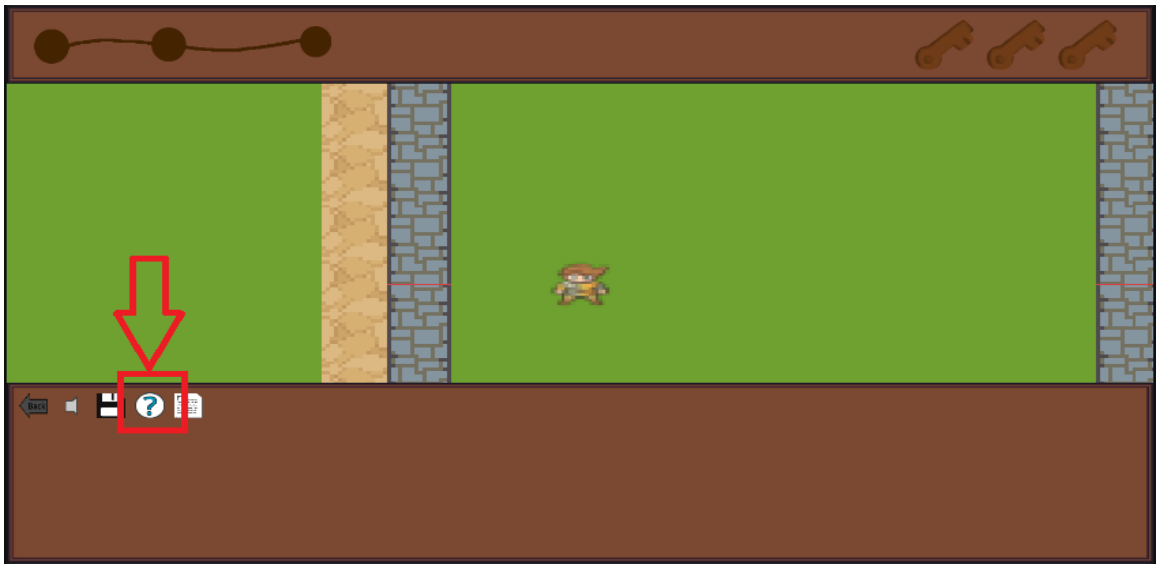
- Once the project loads in Unity Editor, navigate to the Project window at the bottom
- Locate the Scenes folder within the Assets directory
- Double-click the Main_Menu scene
- Click the Play button at the top center of the Unity Editor to run the game
- Use WASD for movement through the scene and E to interact
- Click the Stop button to stop the scene and return to edit mode

11 Sample Scenarios

[Start Game] > [Press New Game]

[1] This is the Main Menu screen and is the first thing that appears to the user after launching the game.

> [Press ‘Controls’ Icon]

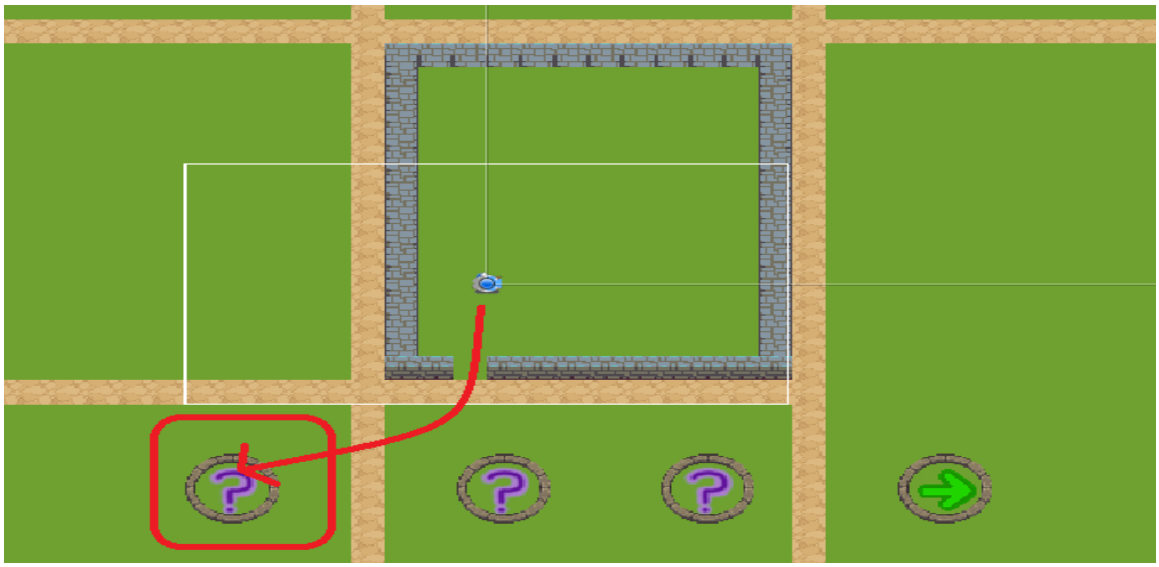


[2] This is the starting point for the game. A game map has been loaded and the player character stands at the center of the screen. Use WASD to navigate and E to interact.

[Press X icon in top right of new window to close]

[3] This is the controls screen to inform the player of how to interact with the game.

[Navigate downwards towards ‘Puzzle_1’ Entrance]

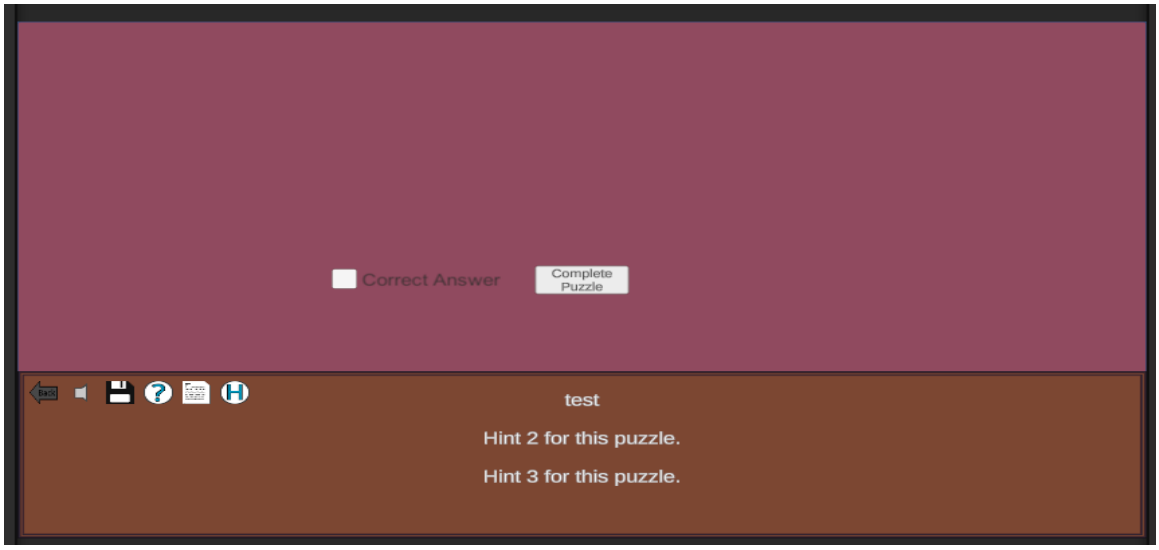


[4] This is a large overview of the game map to indicate where to navigate the player character to continue.

[Stand on Icon and Press E]

- The ‘Interact prompt’ is displayed on the HUD once the player character is sufficiently in contact with the ‘Puzzle Entrance’ Icon.

[Press ‘Complete Puzzle’ button 4 times to display all hints]



[5] This is the placeholder for a puzzle.

[Check ‘Correct Answer’ check box and press ‘Complete Puzzle’ button]

[6] The game has returned the player to the map at the entrance point of the puzzle and a ‘Key’ has been earned by completing the puzzle.

12 References

Start of your text.

[7] D. Thakore and S. Biswas, “Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks,” Proceedings of IEEE Military Communication, Atlantic City, October 2005.

13 Point of Contact

For further information regarding this document and project, please contact **Prof. Daly** at University of Massachusetts Lowell (james_daly at uml.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.