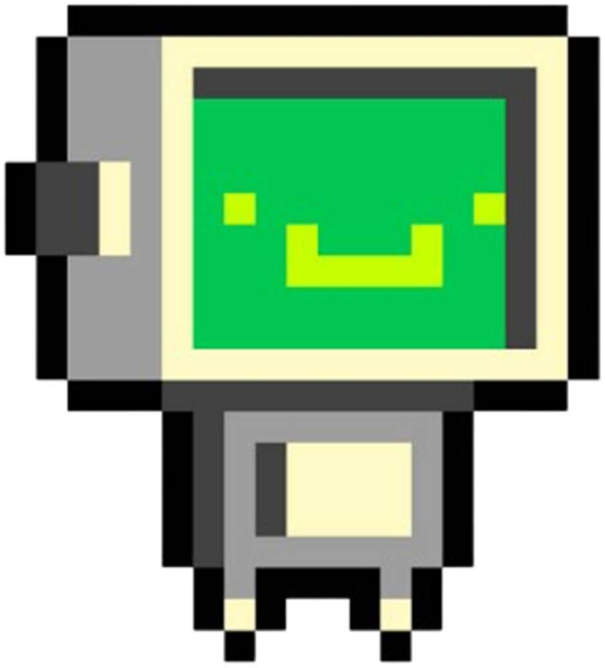
Alan’s Coding Journey

Game Manual



Contents

Background and Overview

In this document, I will be describing the Background and Overview, Requirements, Design, Installation, System Instructions, Sample Sessions, Modifications, and Troubleshooting of Alan's Coding Journey application that is the basis of my Game Development project. Specifically, within this section, I am going to be explaining the background of my domain and an overview of my project. This includes the what the main purpose of my game, the scope of the program, the target users, and assumptions made about what the user should already know.

In the modern world, learning at least one programming language has become an essential skill in today's world. Not only will learning computer programming assist with professional opportunities, but early exposure to coding concepts can significantly improve children's problem-solving abilities, logical thinking, and creativity. However, traditional programming education often struggles to engage young learners due to its abstract nature and complexity.

Computer programming education has evolved significantly over the past decades. Traditional teaching methods often struggled with student engagement and retention due to abstract concepts not yet fully understood and complex syntax rules. However, research has shown that interactive learning through gamification, especially for younger individuals, can significantly improve knowledge retention. Based on multiple studies, students learn programming concepts more effectively when presented in a visual, interactive format that provides immediate feedback and practical application.

Alan's Coding Journey is an educational side-scrolling desktop application designed to introduce young adults to coding concepts through an interactive adventure. The player must

utilize problem solving skills along with side scrolling movements to progress through the game. The main purpose for this game is to introduce fundamental concepts of the C programming language through answering questions and defeating enemies.

Players take on the role of Alan (Alan Turing), a young computer who journeys through four distinct eras (levels) of computer programming history, each representing a different programming language. The journey begins with a tutorial that explains the purpose of the game and teaches the player how to navigate through the levels, then transitions into the basic concepts and syntax of C, progresses through the realm of control structures and functions, explores the world of data handling and arrays, and concludes in the environment of advances programming concepts.

Each of the four levels focuses on core programming concepts appropriate for high school and college students wanting to learn computer science. Players advance by solving coding challenges and defeating enemies using helpful hints and short readings. The difficulty of the questions and reading increases gradually, building on previously learned concepts to allow for a challenge to keep the player entertained.

This product can serve as an educational tool for:

- Students beginning their programming journey
- Teachers introducing coding concepts in a classroom
- Parents trying to find an engaging way to expose their children to programming
- After-school programs focused on non-conventional education

Users should have computer literacy skills, including the ability to use a keyboard and mouse, read at a middle-school level, and perform basic arithmetic. No prior programming experience is required, as the game introduces all concepts from fundamental principles.

The game's scope focuses on:

- Basic syntax and structure
- Core concepts (variables, conditional statements, loops, functions)
- Progressive skill development by increasing difficulty throughout three encounters per level
- Interactive learning through immediate feedback

Through its mix of entertainment and education, Alan's Coding Journey makes coding accessible and enjoyable for beginner learners, laying the foundation for future developers!

Requirements

This section details the requirements of Alan's Coding Journey which are taken directly from the GameMaker Help Centre resource. The details listed below will include the hardware and software requirements for not only the user but for the developer, the goals for the user, and the main functionalities of the system.

Hardware and Software Requirements:

For Users:

The game requires a computer system with the following minimum specifications to play:

- Operating System: Windows 10/11
- Memory: 4 GB RAM
- Graphics: Dedicated graphics card
- Storage: 500 MB available storage
- Display Resolution: 1280x720 or higher
- Input Devices: Keyboard and Mouse
- Internet Connection: Not required (offline game)
- Workbook: C Programming Workbook

For Developers:

The development of the game requires a computer system with the following minimum specifications:

- Main IDE: GameMaker Studio 2

- Asset Creation: Graphics software for asset creation (e.g., Krita)
- Operating System: Windows 10/11
- Processor: Quad-Core CPU
- Memory: 8 GB RAM
- Graphics: Dedicated graphics card
- Storage: 3 GB available storage
- Internet Connection: Required
- Version control system: GitHub (optional but recommended)

Goals for User:

The primary purpose of Alan's Coding Journey is to provide a fun introduction to programming concepts and languages for young/unexperienced individuals wanting to learn. The game aims to provide the user with an experience where should be able to:

- Learn fundamental programming concepts starting from an introduction and going up to OOP.
- Understand the basic syntax and structure of C.
- Progress through increasingly complex programming challenges and be able to use lessons used to answer questions previously without the need to go back.
- Feel a sense of familiarity with C.
- Save and resume their educational journey at any time to allow for learning at own pace.

Product Functionalities:

Core Gameplay:

1. Multi-level progression with multiple encounters in each level depending on the depth of content for each section.
2. Side-Scrolling character movement with platformer jumping mechanics.
3. Two-part enemy encounters requiring coding question and a short fight to overcome.
4. Progressive difficulty increases across each level and throughout each section.

Educational Features:

1. Interactive coding challenges with immediate feedback.
2. In-game programming language tutorials for new concepts based on Dr. Redfield's book chapters.
    a. Expandable information panels for detailed explanations.
3. Randomized questions and answers for each encounter.

Progress Management:

1. Automatic progress saving after completion of level.
2. Level unlocking system based on completion.
3. Health system with multiple lives and flashing animation when hit or question answered incorrectly.
4. Trophy system with user gaining trophies based on level completions.
5. Start menu with new game and load game options.
6. Level selection screen with clear progression path.
7. Animated game-over screen.
8. Navigation between tutorial and encounter sections.
9. Visual feedback for fights and correct/incorrect answers.

These functionalities work together to create an educational experience that maintains engagement while steadily building programming knowledge through relevant questions and fun gameplay.

Design

This section goes over the design (how the game was broken down and created) of Alan's Coding Journey. The details listed below will include the reasoning behind the selection of the languages and OS, how data is managed and organized, the system flowchart and sequence of actions, and how the main screens look to the user.

Languages and Operating System:

This project was developed using GameMaker Language (GML) and runs on Windows operating systems. GML was selected as it is the native programming language in the GameMaker Studio 2 IDE, however, it was easy to understand and use as it is like JavaScript and C languages which I have previous experience in. Its simplified syntax and built-in functions allowed me to develop the functionalities of each element and sprite while making it really on user input. Windows was chosen as the target platform as that is the operating system of my desktop and with my free GameMaker account, I can only build the installer for operating systems that the game was created on. On top of that. Most educational resources that schools provide are Windows-based so the installation and use of the .exe file should be very simple.

Data Structures:

Although no database was directly used, the game utilizes data storage through a save file system, maintaining player progress across sessions. Game data is structured using arrays and global variables, with save data stored in JSON format.

For the save gave functionality, the game stored the highest level the user has reached in a file called savedata.sav. It also saves if the game has been completed. This file is stored in the local AppData folder, and the contents can be viewed below in *Figure 1*.

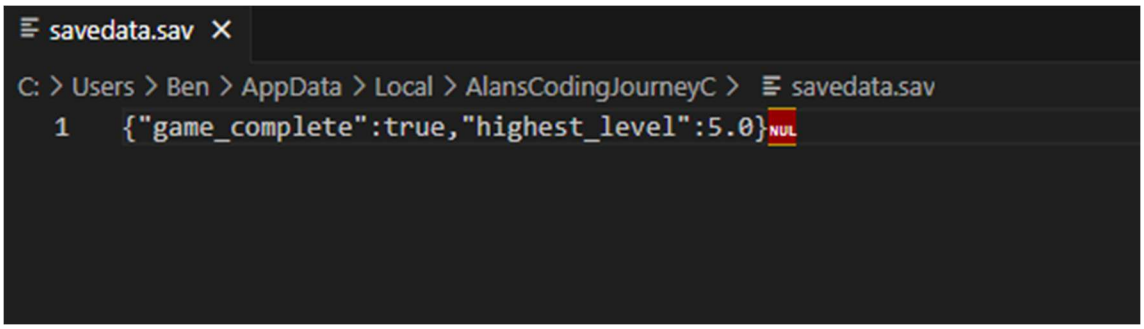


Figure 1: Data stored in local save file

For the data structures like the questions strings and answers, the game stores these in the local modules where they are necessary. Specifically, in the creation code of the objects as seem in *Figure 2* down below where we can see how the questions are stored in the question panel object.

```
// Tutorial Question
questions[0, 0] = "What is your name?\n (Hint: It is the name of the game!)";

//Level 1
// Level 1, Section 0: Introduction and Main
questions[1, 0] =
[
    [   // First Question
        "How many bytes are in a KB?", // First variant
        "How many bytes are in a MB?", // Second variant
        "How many bytes are in a GB?" // Third variant
    ],
    [   // Second Question
        "What part of a computer manipulates and stores data?", // First variant
        "What part of a computer processes data?", // Second variant
        "What part of a computer receives input?" // Third variant
    ],
    [   // Third Question
        "The third phase of software engineering that we are focusing on is _______.", // First variant
        "The physical parts of the computer are called _______.", // Second variant
        "What type of language is C?" // Third variant
    ],
    [   // Fourth Question
        "How many bits are in a byte?", // First variant
        "When running, every C program has and starts executing at the _______.", // Second variant
        "What part of a computer presents output?" // Third variant
    ]
];

// Level 1, Section 1: Variables and Expressions
questions[1, 1] =
[
    [   // First Question
        "In C, the syntax to declare an integer variable called x is", // First variant
        "In C, the syntax to declare a decimal variable called num is", // Second variant
        "In C, the syntax to declare a character variable called c is" // Third variant
    ],
    [   // Second Question
        "Which header file should be included to get access to input and output functions?", // First variant
        "Which is the C command to show text on the screen/monitor?", // Second variant
        "What place holder in a format string will print out a double/decimal value?" // Third variant
    ],
    [   // Third Question
        "In order to read in and manipulate data, we need _______ to store values.", // First variant
        "Each statement in a C program should end with what?", // Second variant
        "Which statement is a complete assignment of an integer value to a variable called x?" // Third variant
    ]
];
```

Figure 2: Question strings stored in the obj_question_panel.

Major Code Modules:

Although the game has many objects and rooms, the game is broken down into are broken up 4 main modules or components. These are the:

1. Game Controller Module: Manages game state, save/load functionality, and global variables. Maintains persistence across room transitions and manages the level progression logic.
2. Level Controller Module: Controls level-specific behavior such as enemy spawning, collision detection, and victory or death conditions. Manages the transition between learning and encounter phases.

3. UI Controller Module: Handles user interface elements, including menu systems, button interactions, and information panel display. Manages the presentation of educational content and user interface.
4. Player Controller Module: Manages player character movement, animation states, collision responses, and boundary checking. Handles interaction with game elements.
5. Educational Content Module: Stores and manages the programming challenges, correct answer checking, and educational content information. Controls the progression of learning material and validation of user responses.

System Flowchart:

The game follows a hierarchical structure where players progress through increasingly complex programming concepts as we can see in *Figure 3* below:

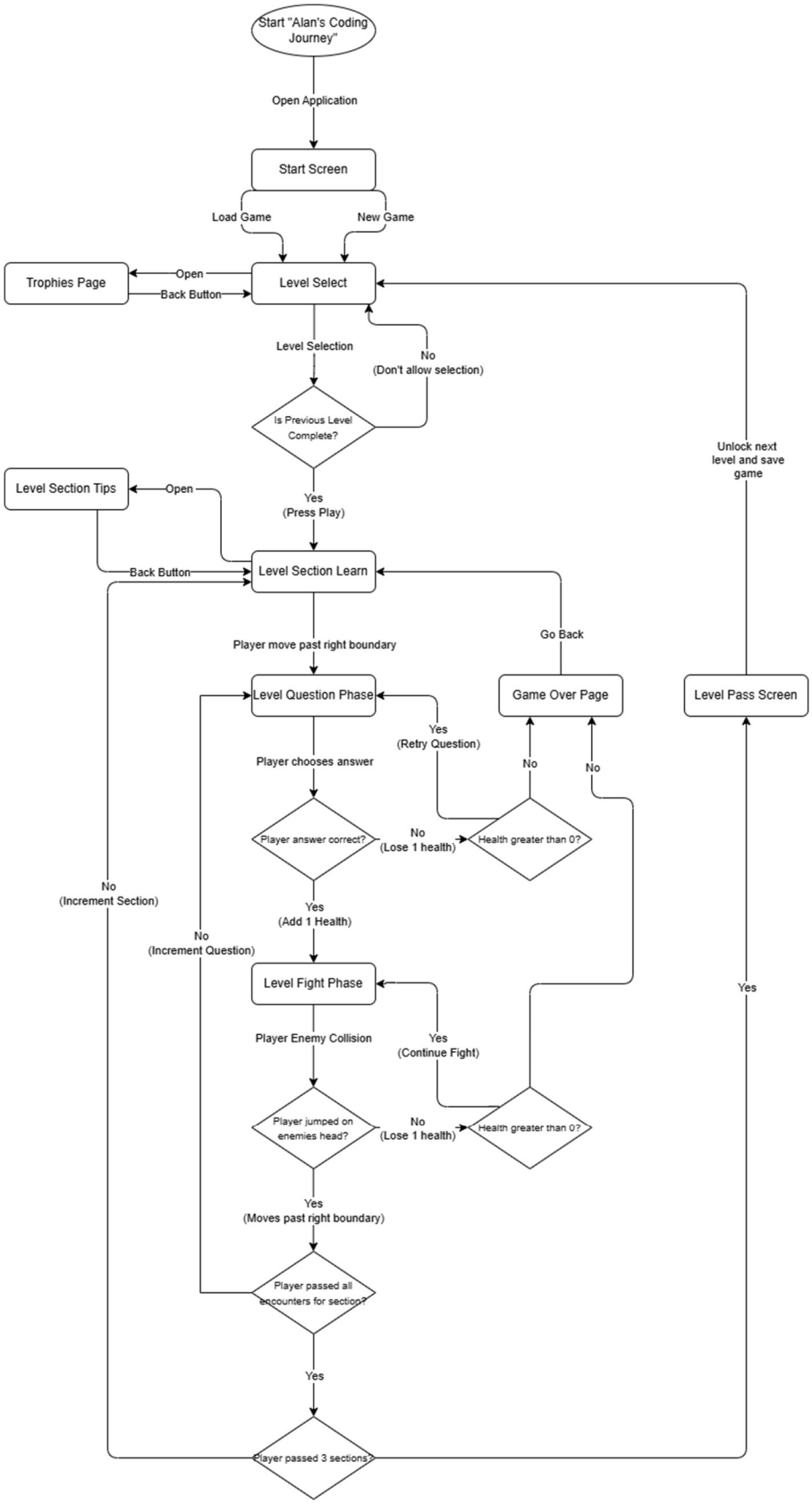


Figure 3: System Flowchart

To elaborate a little further, the boxes in flowchart above show the major screens of the game. They are the:

- Start Screen
- Level Select Screen
- Trophies Page Screen
- Level Pass Screen
- Game-Over Screen
- Level Section Learn Screen (Unique for each of the 3 Sections for the 4 Levels and the Tutorial Level)
- Level Section Tips Screen (Unique for each of the 3 Sections for each of the 4 Levels and the Tutorial Level)
- Level Question Phase Screen (3 unique possibilities for each of the encounters for each of the 4 Levels and Tutorial Level)
- Level Fight Phase Screen (Unique for each of the 4 Levels and the Tutorial Level)

Interface Design:

The game presents a consistent user interface across all screens. However, the key screens are:

1. Start Screen: Features the game title, animated character, and options for "New Game" and "Load Game". The dark background with binary aesthetic establishes the programming theme. (*Figure 4* Below)

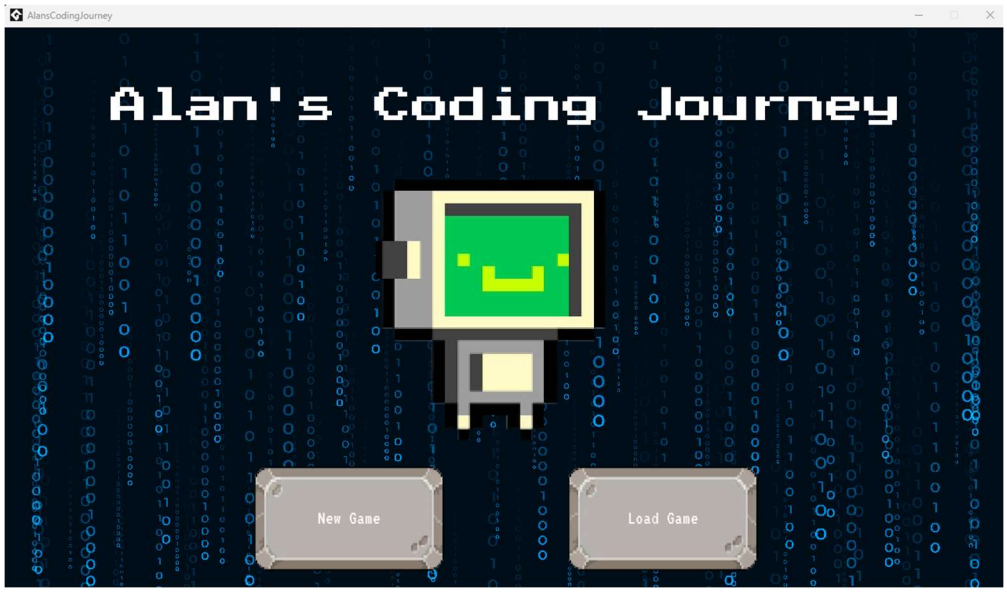


Figure 4: Start Screen

2. Level Select Screen: Displays available levels in a horizontal progression, with locked levels clearly indicated. Each level shows its associated concept icon and name. The user can click the red back arrow in the top left corner to return to the Learn Screen. (*Figure 5* Below)

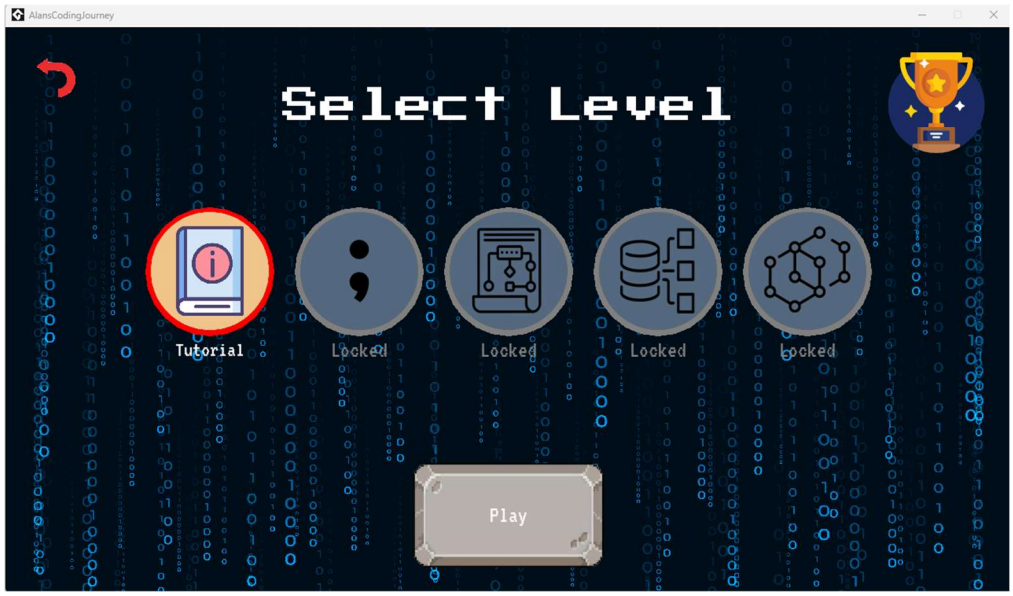


Figure 5: Level Select Screen

3. Learn/ Level Encounter Tips Screens: User is presented with the background of the level with a chalkboard that allows for user selection for helpful hints in the Level Section Learn Screen. The player's character and hearts remain visible in the game world during this screen. If the user clicks on the board, the Level Section Tips Screen will be displayed which is an expandable blackboard panel that displays relevant information for the upcoming encounters for each section. The information in the Level Section Tips screen is a compressed version of the content from the relevant chapter within Dr. Redfield's book. The user can then click the red back arrow in the top right corner to return to the Learn Screen. (*Figure 6 & Figure 7* Below)

Figure 6: Level 1 (Fundamentals) Section 1 (Introduction and Main) Learn Screen

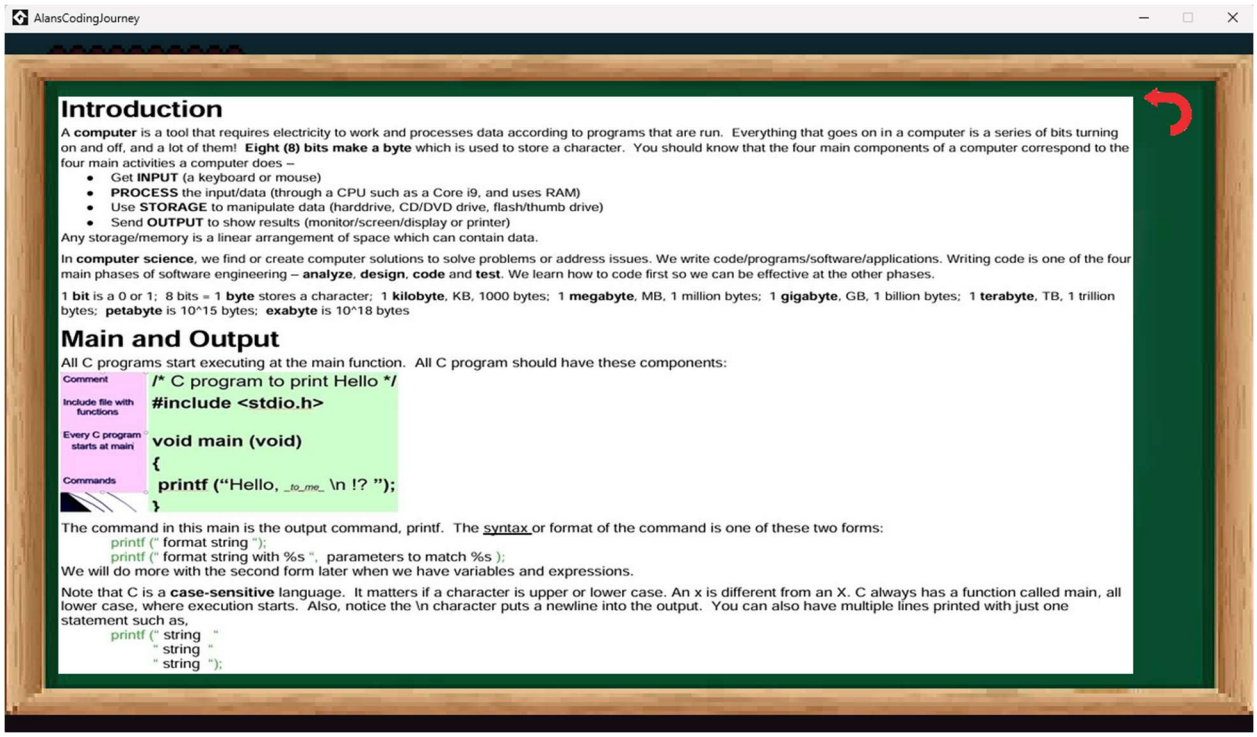


Figure 7: Level 1 (Fundamentals) Section 1 (Introduction and Main) Tips Screen

4. *Level Question/Fight Encounter Screens*: Combines platforming elements with a programming related challenge. For the Level Question Encounter, the screen displays:
    a. Health indicator (top left)
    b. Coding challenge question (center top)
    c. 3 Multiple choice answer buttons
    d. Player character and enemies
    e. Background elements

If user selects correct answer, the user will get a message saying that their choice was correct or incorrect, and if correct, that the movement is enabled, which after pressing okay, will fade out the question panel and buttons which transitions into the Level Fight Encounter Screen. (*Figures 8 & 9* Below)

Figure 8: Level 1 (Fundamentals) Section 1 (Introduction and Main) Question 1 Screen

Figure 9: Level 1 (Fundamentals) Fight Phase Screen

5. Trophy Screen: The screen displays “Trophies” at the top of the screen. To the left of the name, is a red arrow button. There are 6 achievable trophies in the middle. Not all trophies are unlocked. There are two lines of text underneath each image. The first gives the name of the trophy. The second states how the trophy is unlocked. Once a user completes each level, they will unlock the image and text for the trophy. The user can click the red back arrow in the top left corner to return to the Level Select Screen. (*Figure 10* Below)

Figure 10: Trophies Screen

Installation

This section goes over how to acquire and play Alan's Coding Journey. The guide below details how to install, setup and get access for not only the user, but for developers aiming to work on the game. The project structure is also within this section.

For Users:

The installation process for Alan's Coding Journey has been streamlined into a single Windows executable file for ease of use (*Figure 10* Below). Upon downloading the installer, users should follow these steps:

1. Locate the downloaded installer file (AlansCodingJourneyInstaller.exe).
2. Double-click the installer to begin the installation process.
3. Follow the on-screen prompts to select installation location and create desktop shortcuts.
4. Once installation completes, launch the game through the created desktop shortcut or start menu entry. (*Figure 11* Below)

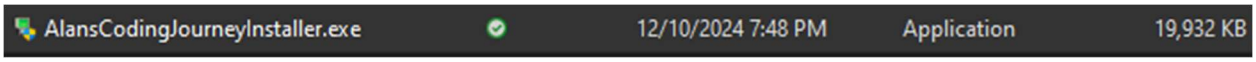


Figure 10: Alan's Coding Journey Installer

Figure 11: Alan's Coding Journey Desktop Shortcut

Currently the installer is only acquired through shared links and private messages as the game is still a demo, however, once ready for complete deployment, the game will be available for download on a unique website.

For Developers:

The complete source code for Alan's Coding Journey is maintained in a public GitHub repository which can be accessed by going to https://github.com/bebadinia/AlansCodingJourneyV2/. To begin development:

1. Set up development environment.
    a. Go to https://gamemaker.io/en.
    b. Click on “Download” Button in the top right corner.
    c. Find and open the installer.
    d. Follow the installation steps and open Gamemaker 2 Studio IDE.
2. Get access to source code.
    a. Go to GitHub repository. (*Figure 12* Below)
    b. Clone the repository, either by downloading as a .zip or by using Git.
    c. Open the project file (AlansCodingJourneyC.yyp) in GameMaker Studio 2.

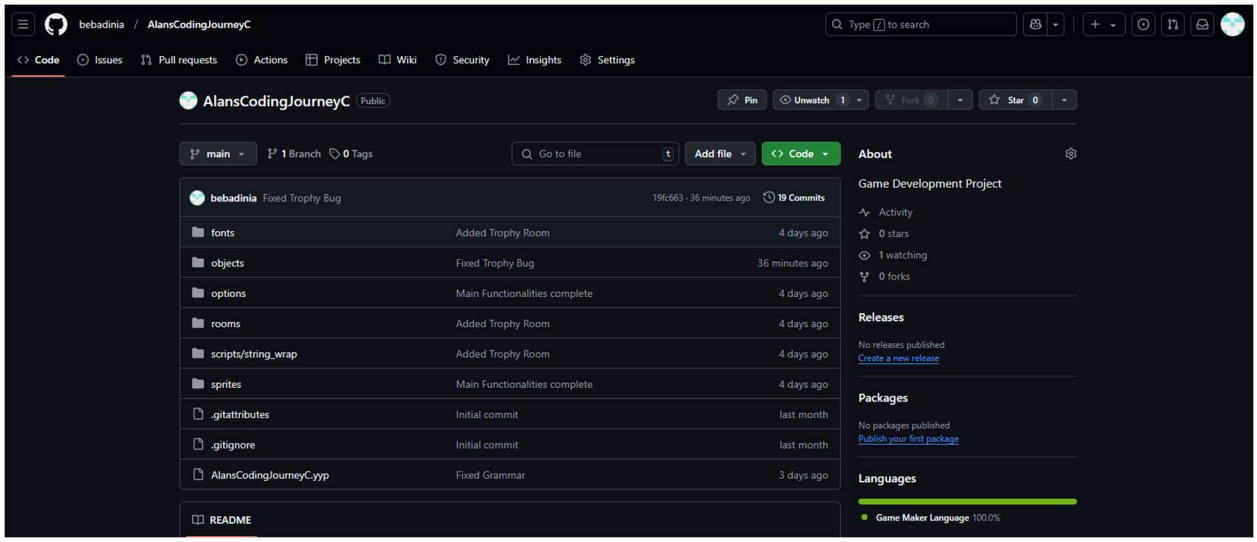


Figure 12: Public GitHub repository

Project Structure:

The source code follows a structured organization within the GameMaker project. This folder contains:

- Fonts Directory: contains the game's unique fonts for all elements.
- Objects Directory: contains the game's core functionality through organized controller character, and user interface objects.
- Sprites Directory: contains all visual assets, including character animations, interface elements, and environmental graphics.
- Rooms Directory: contains the game's various screens and levels, from menu interfaces to gameplay environments. Essentially connects all the elements above together.

The repository includes all necessary project files and assets required for development. No additional external dependencies are required beyond GameMaker Studio 2.

Using the System

This section covers the steps needed to use the Alan's Coding Journey system. The guide below details how to start and play the game after installation, while detailing the actions that the user needs to do to fully enjoy the game.

Getting Started:

After installation, launch Alan's Coding Journey by double-clicking the desktop shortcut or selecting it from the installation file. The game opens to a welcoming start screen with the name of the game featuring an animated Alan sprite and two options: "New Game" and "Load Game." For first-time players, select "New Game" to begin your programming adventure.

Game Progression:

This game is completely single player and can save the players progress to be able to return to. The game guides you through a natural progression of programming concepts across different languages with each level having its own enemy and backdrop, reflecting the technological context and enemy of its concept. The progression of levels is as follows:

1. Tutorial Level: Cherry Blossom Forest Pass
2. Fundamental Level: 8-bit Grassy Path; Bug Enemy
3. Control Flows Level: Rocky Waterfall Path; Virus Enemy
4. Data Structures Level: Dark Swamp Path; Corrupted USB Enemy
5. Complex Concepts Level: Rainforest Path; Trojan Horse Enemy

So based on the sequence above, the player will start with the tutorial where they will learn how to play the game, then go to the 1st level, where the most fundamental concepts are

introduced. As you complete each level, the next levels becomes available, building upon previously learned concepts.

Basic Controls:

Navigate through the game using standard controls left-and-right arrow keys for movement and the spacebar for jumping. During question challenges, use your mouse to select one of the three possible answers which are formatted as buttons. The game then alternates between platforming sections and question encounters.

Navigating Levels:

Each level consists of three sections and each section consists of one or more question encounters depending on the amount of information in each chapter. Before each section, you'll enter a learning phase where information about the upcoming encounter is introduced. Click on the information panel to expand it and study the material at your own pace. Once ready, progress through the right boundary to face the question challenges for the section.

Handling Encounters:

During encounters, you'll face enemies that can only be defeated by correctly answering programming questions. Read each question carefully and select your answer from the provided options. Correct answers allow you to progress to the fight with an increase of a heart, while incorrect answers result in the loss of a heart. During the fight, the user will have to jump on the enemy sprites head while trying to avoid being hit. Every hit will result in a loss of a heart and put a 2 second invincibility shield around Alan which shows him flashing. You have five hearts per level attempt, and if the number of hearts ever reaches 0, the user will be redirected to the

Game-Over screen where the user will return to the Level Select screen to retry the level that they failed.

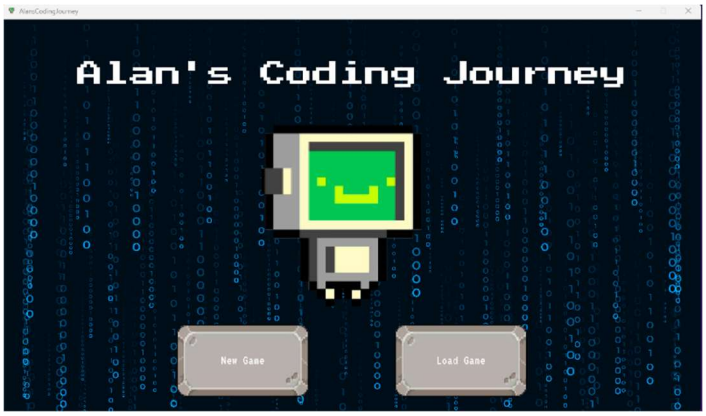
Completing Level:

Successfully completing all three encounters within a level to unlock the next programming language, the Level Pass screen displays informing you of the current level passed along with the next level unlocked. Each victory brings you closer to becoming familiar with various programming concepts, from basic syntax to complex functions. The game automatically saves your progress after completing each level. You can safely exit the game at any time and continue from your last completed level by selecting "Load Game" at the start screen. Your highest unlocked level and trophies are preserved between sessions.

Sample Sessions

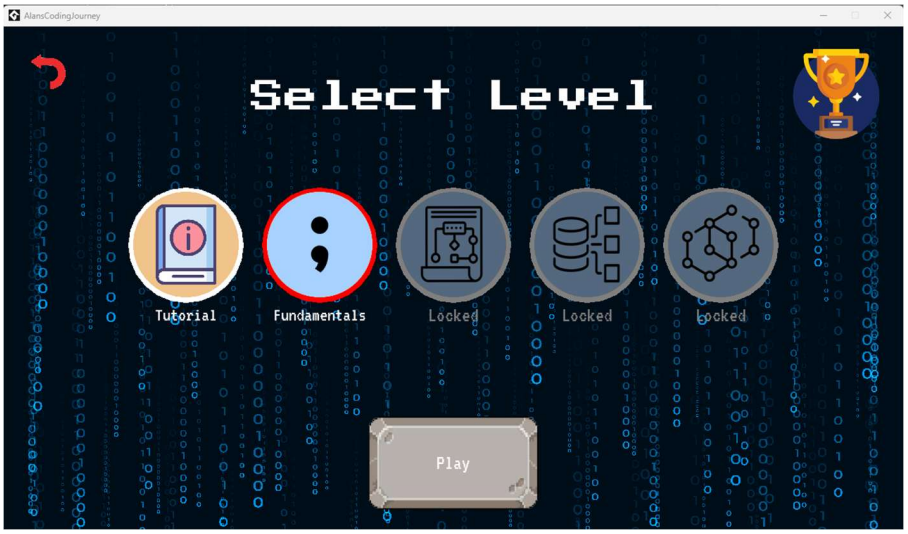
This section covers a sample players gaming session for the Fundamentals level in the Alan's Coding Journey system. For each step or interaction, there will be the name of the step, photo of the screen underneath it, along with a description of each below the photo.

1. **Starting the Game**



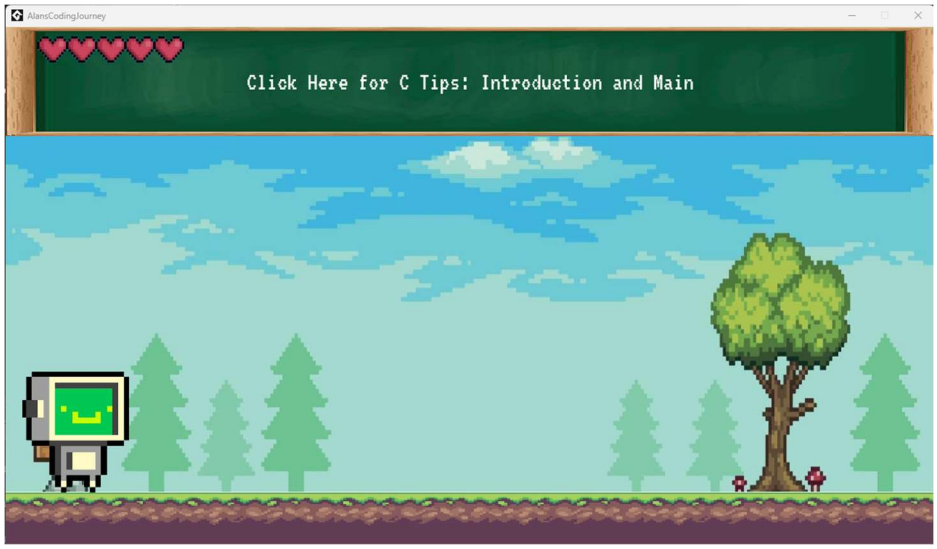
The start screen welcomes players with an animated Alan and two buttons: "New Game" and "Load Game." The dark background features a matrix-style binary pattern, setting the programming theme. Here, we will press the "Load Game" button since we have already completed the tutorial level.

2. **Selecting a Level**



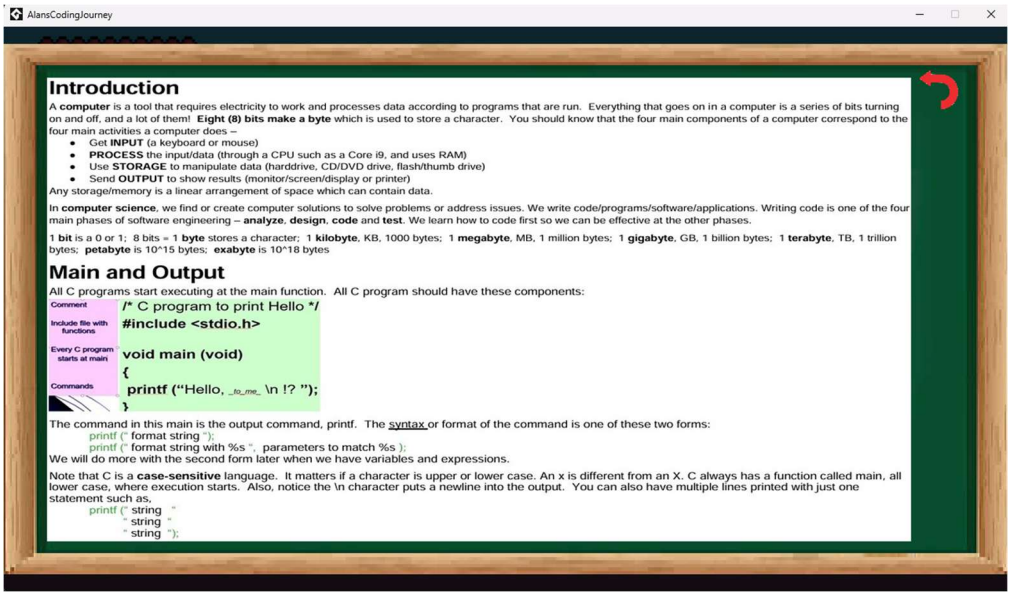
After loading a game, players see the level select screen displaying a tutorial level and 4 C concepts level options. Tutorial is initially unlocked and selected, with subsequent levels locked until prerequisites are completed. From here we select “Fundamentals” and press play.

3. Level 1 (Fundamentals) Section 1 (Introduction and Main) Learn Screen



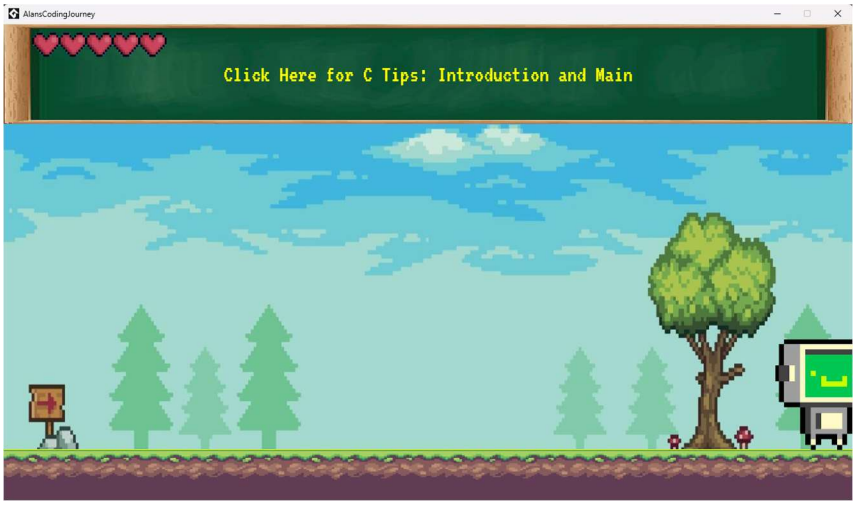
Players enter a learning room featuring Alan, hearts display, and an information panel introducing programming concepts. From here, we will click on the panel to expand it.

4. Level 1 (Fundamentals) Section 1 (Introduction and Main) Tips Screen



Clicking the information panel reveals detailed explanations of the programming concepts for that section, based on the chapter content from Dr. Redfield's Workbook. By reading the panel, we learn about the basics of computer science and Main/Output, which is the building blocks of programming. From we press red arrow in the top right corner, and we return to the Learn Screen.

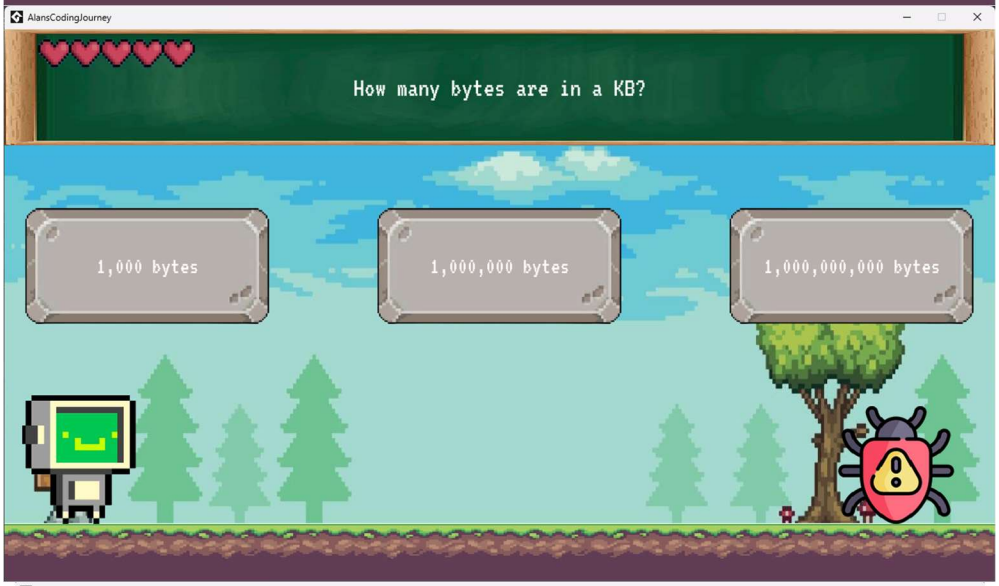
5. **Going to the Encounter Phase**

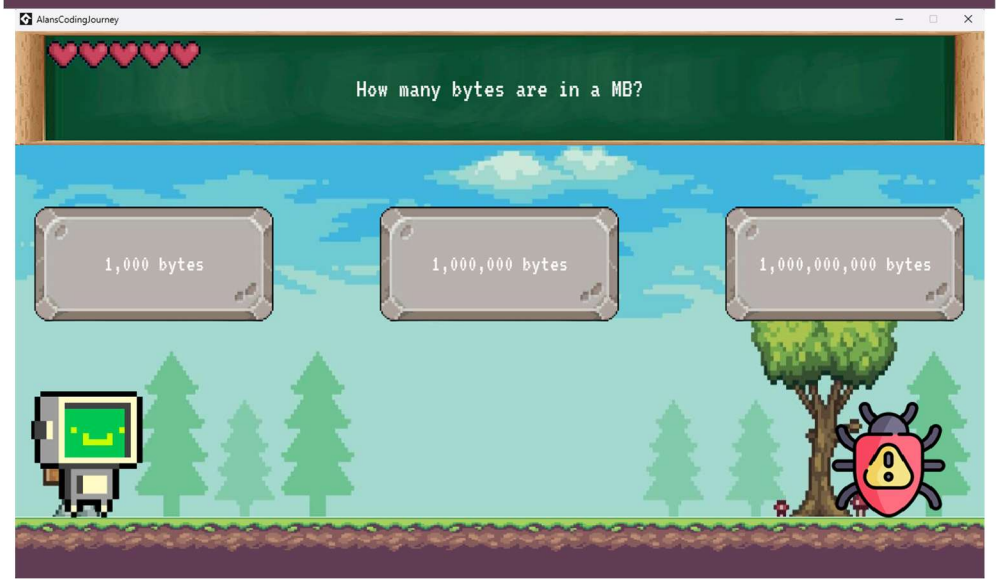


We use our right arrow on our keyboard to move to the right and once we move past the right boundary of the room, the Level Question Phase Screen appears.

6. **Level 1 (Fundamentals) Section 1 (Introduction and Main) Question 1 Screen**

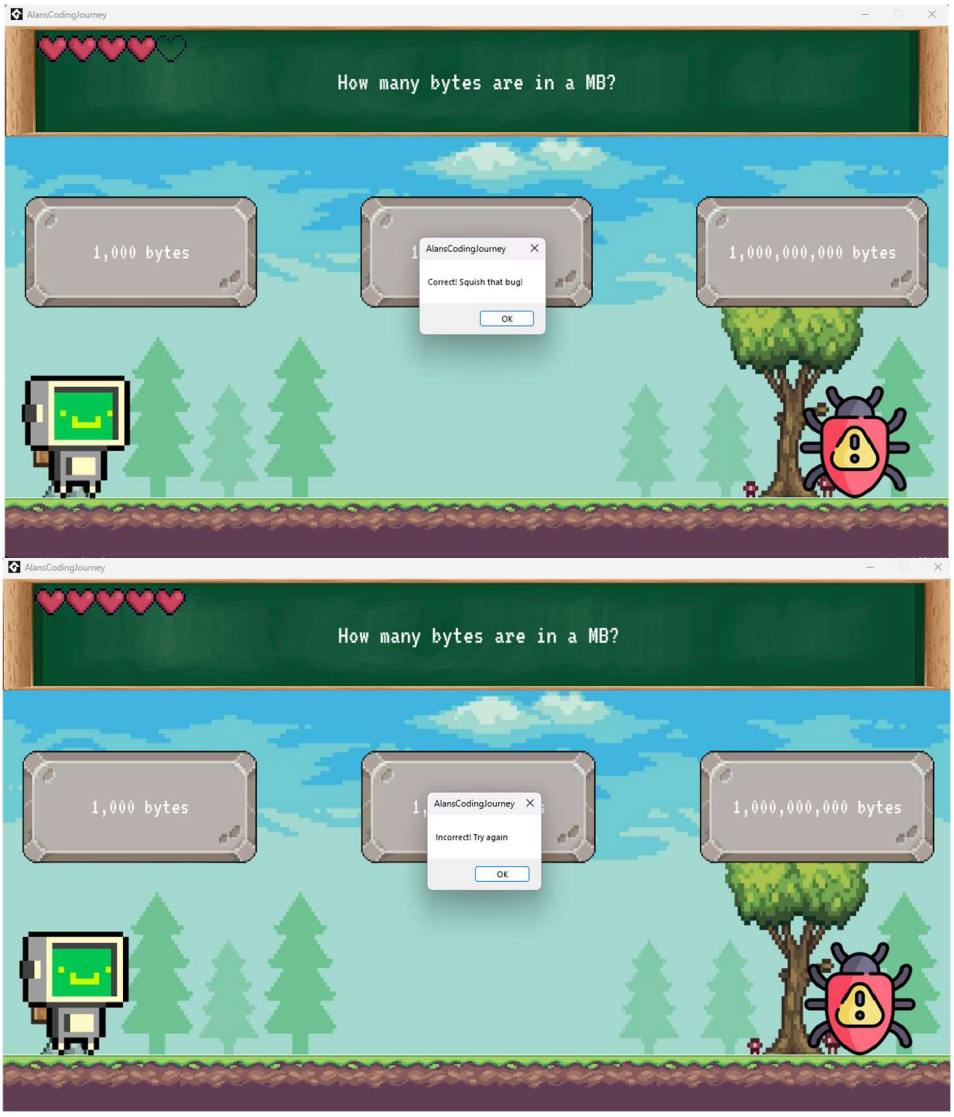

AlansCodingJourney
How many bytes are in a GB?
1,000 bytes
1,000,000 bytes
1,000,000,000 bytes


AlansCodingJourney
How many bytes are in a KB?
1,000 bytes
1,000,000 bytes
1,000,000,000 bytes


AlansCodingJourney
How many bytes are in a MB?
1,000 bytes
1,000,000 bytes
1,000,000,000 bytes

During an encounter, the screen displays a related question at the top, 3 multiple-choice answers, and an enemy that must be defeated. Every question screen contains three sets of possible questions and answers that are randomly selected with every question phase (As seen in the 3 screens above). We can also notice that questions are relevant to the information provided on the previous screen. The movement of both Alan and the enemy are locked until the question is answered correctly.

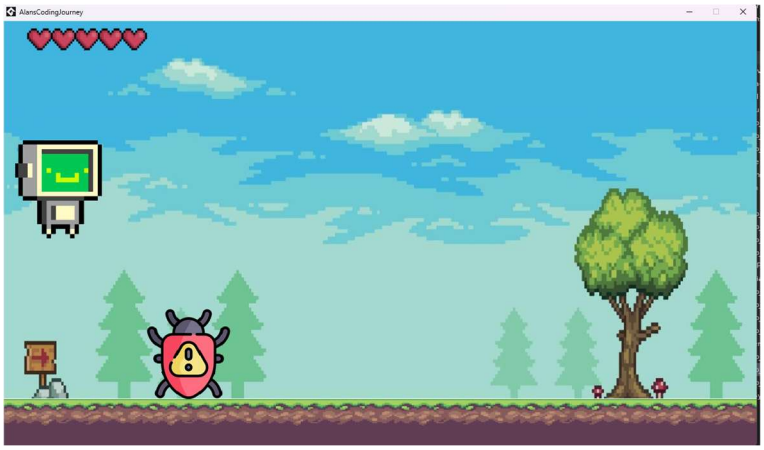
7. Answering the Question



When players select the correct answer, visual feedback shows “Correct! Squish that bug!”, the user regains a hearts when they press “OK”, and the board along with the buttons fade out. However, if the user clicks the wrong choice, we can also see feedback that says “Incorrect! Try

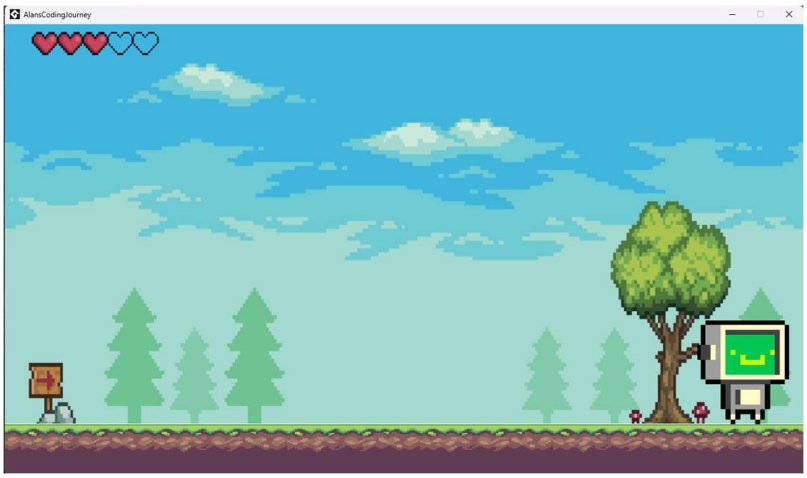
again" while removing a heart. When the correct answer is finally chosen, movement will resume, and you must jump on the bug's head.

8. **Level 1 Fighting Phase Screen**



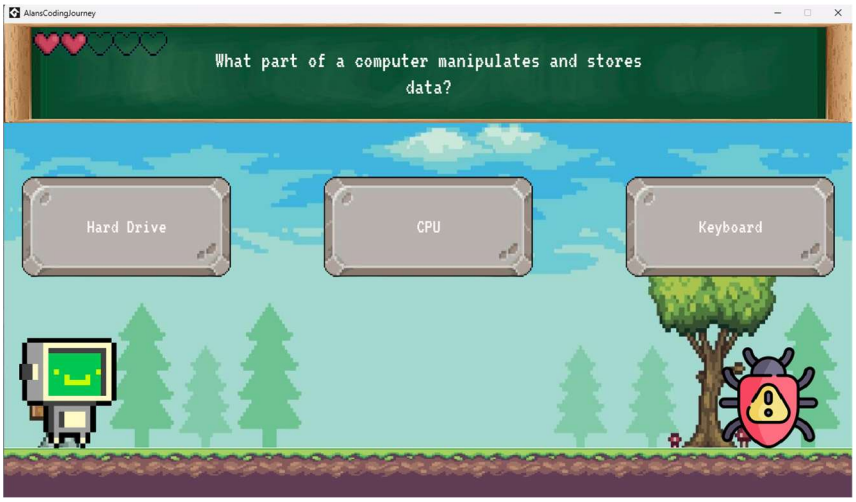
During this phase, the bug will move towards you and try to hit you. If it succeeds in hitting you from the side, it will take a heart away, push you back, and make your character flash to signal invincibility to damage for 2 seconds to allow time to reset. During this time, the enemy cannot hurt you and you can't defeat it as well. Once you turn back into a solid color once more, you must attempt to move with the right and left arrows to try to jump on its head. With each section to reach, the enemy will be faster and faster to kill.

9. **Going to Next Phase**



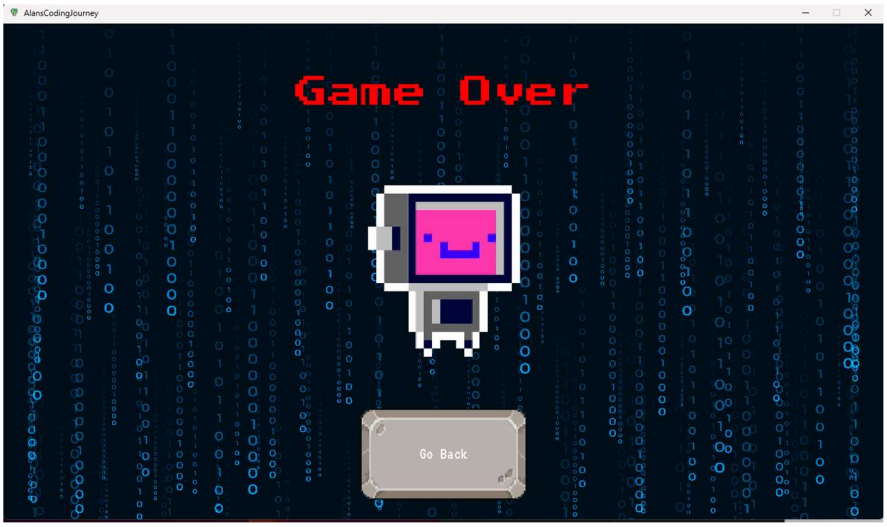
Once you do, the enemy fades away and you can move to the right room boundary, which was previously locked, to continue. This will take you to the next encounter.

10. Level 1 (Fundamentals) Section 1 (Introduction and Main) Question 2 Screen



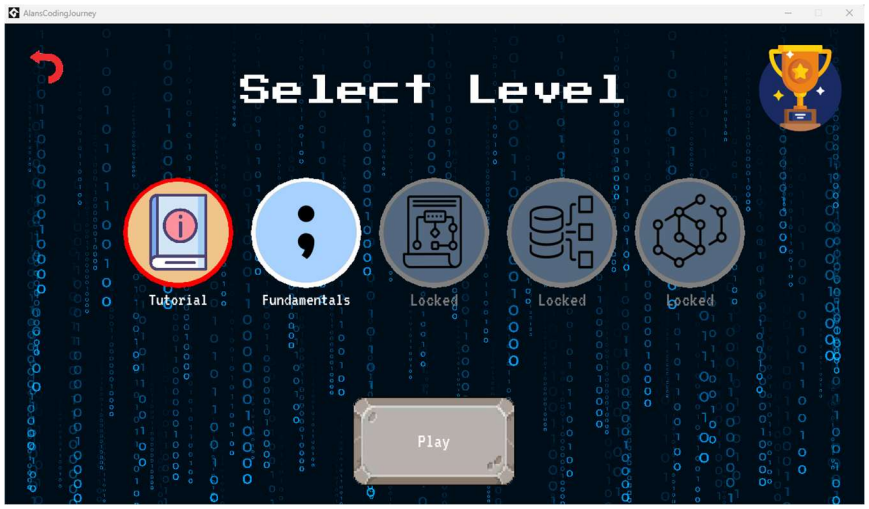
This will take you to the next Question phase for the section. We can see here that the hearts remain after each encounter and only reset once the level is complete, or Alan dies. Once again, we must select an Answer for the question. Unfortunately for us, we answered the question incorrectly twice which brings our hearts down to 0.

11. Game-Over Screen



If Alan's heart ever reaches zero, then an animation will play where Alan falls, and it shows "Game Over". From here the user presses the "Go Back" which returns the user back to the "Level Select" screen.

12. Level Select Screen



Back at the Level select screen, we can see that we did not unlock the next level and have to retry the fundamentals level if we want to continue. We can also press the trophies button in the top right of the screen.

13. Trophies Screen

After clicking the trophy icon, we are taken to our trophy page. This shows us our rewards for what we have completed along with the challenges we have yet to complete.

Modifications

This section goes over the future work and improvements that can be made to Alan's Coding Journey. The details listed below will include instructions on how to make changes to keep formatting consistent and future work that I have planned.

Code Modification Instructions

To make changes to the code, the user will need to go back up to the Installation For Developers section to properly install and view the code. However, only the project is loaded in GameMaker 2 Studio IDE, the user can should make changes using the following conventions to maintain organization and readability.

Convention 1: All object names begin with "obj_" followed by their function (e.g., obj_game_controller, obj_learn_panel). Room names follow a similar pattern, starting with "rm_" and indicating their purpose. The same goes for sprints with "spr_". These naming conventions should be kept when making modifications.

Convention 2: For visual modifications, sprites are organized in the asset browser according to their function. Character animations have consistent frame counts and dimensions. Any new animations should use these specifications to maintain visual consistency.

Convention 3: For overall organization, each type of element is in a folder with similar types. For example, all the objects are in one big folder called "Objects". Furthermore, each element folder is broken down into groups based on rooms and types.

Future Enhancements

The current version of Alan's Coding Journey could be enhanced through several additions. The first thing being increasing number of levels to allow the user to properly learn the material. This way, the game is not so short, and players can experience a variety of topics and questions. Another main functionality to add would be adding sound effects and background music. This would also come with the addition of a settings menu with does not currently come with the game.

To create a bigger challenge for experienced players can also be by adding different moves and adding a health bar for each enemy. This way, the enemies don't die after a single jump and challenge increases as user's become more advanced.

Another enhancement could be adding the original idea of allowing players to write simple programs within the game environment using interactive code editing segments. This would provide hands-on experience with actual programming.

I believe also adding an option for teachers to create custom questions and answers sets for their students could be very beneficial for education in different areas.

I think another interesting modification is adding more trophies that would provide additional motivation for players to complete specific challenges. This would recognize and reward players with visual upgrades for not only level completion but for things like perfect runs and speed completions.

Future versions could also incorporate difficulty adjustments based on players' coding knowledge. This system would modify question complexity and put a timer for question encounters.

A tracking or grading system could be implemented to help players or educators identify areas where more practice might be beneficial.

As this version of the game was created for Dr. Redfield's CS1311 Data Structures class, I hope to continue working on this game on by applying some of these functionalities to make the game more robust.

Troubleshooting

This section goes over the troubleshooting that can still occur during a user play-through of Alan's Coding Journey. The table below gives the symptoms, causes, and solutions to these issues/bugs.

Symptom	Cause	Solution
Game fails to launch.	1. Insufficient system resources. 2. Installation files corrupted or missing. 3. Windows compatibility issue.	1. Close other applications or free up space. 2. Uninstall and reinstall the game. 3. Right-click the game executable, select Properties, and enable "Run in compatibility mode for Windows 10/11".
Save file not loading.	Corrupted save data.	Go to the save file location and delete 'savedata.sav'.
Controls become unresponsive.	1. Multiple input devices connected or system input conflicts. 2. Windows input service issue.	1. Disconnect additional input devices and restart the game. 2. Restart Windows Input Service.
Black screen on startup.	Display resolution mismatch or graphics driver issue.	1. Check Display Settings in Windows, ensure resolution matches game requirements. 2. Update graphics drivers through Device Manager
Error message: "File not found".	Missing or corrupted game assets.	Verify game files through the installer. If issue continues, uninstall and reinstall the game.
Learning panels fail to display.	Temporary UI rendering issue.	Close and restart the game.
Game runs slowly	Background processes consuming resources	Close unnecessary applications in Task Manager

References

[1] “5 Benefits of Gamification.” *Smithsonian Science Education Center*, 10 Mar. 2016, ssec.si.edu/stemvisions-blog/5-benefits-gamification. Accessed 8 Jan. 2025.

[2] “C++ Exercises - C++ Practice Set with Solutions.” GeeksforGeeks, 23 Sep. 2024, www.geeksforgeeks.org/cpp-exercises/. Accessed 11 Dec. 2024.

[3] Ebadinia, Ben. “AlansCodingJourneyV2: Game Development Project.” GitHub, 7 Dec. 2024, github.com/bebadinia/AlansCodingJourneyV2/. Accessed 11 Dec. 2024.

[4] Mathatoo, Gurpreet S, and Sam Spade. “How to Make Buttons in GameMaker.” GameMaker, 27 Mar. 2023, gamemaker.io/en/tutorials/how-to-make-buttons.

[5] Novak, Jeannie. Game Development Essentials: An Introduction. Delmar Cengage Learning, 2012. Accessed 11 Dec. 2024.

[6] Pinto, Mário, and Teresa Terroso. *Learning Computer Programming: A Gamified Approach*, Dagstuhl Publishing, 11 July 2022, drops.dagstuhl.de/storage/01oasics/oasics-vol102-icpec2022/OASIcs.ICPEC.2022.11/OASIcs.ICPEC.2022.11.pdf. Accessed 8 Jan. 2025.

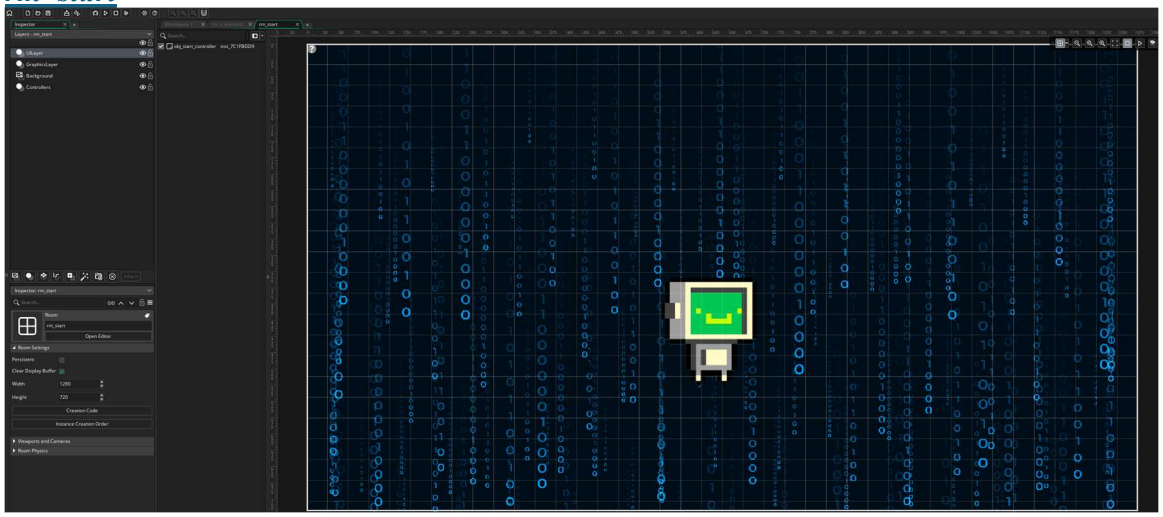
[7] Redfield, Carol Luckhardt. *C Programming Workbook*. St. Mary’s University, 2025. Accessed 8 Jan. 2025.

Appendix A

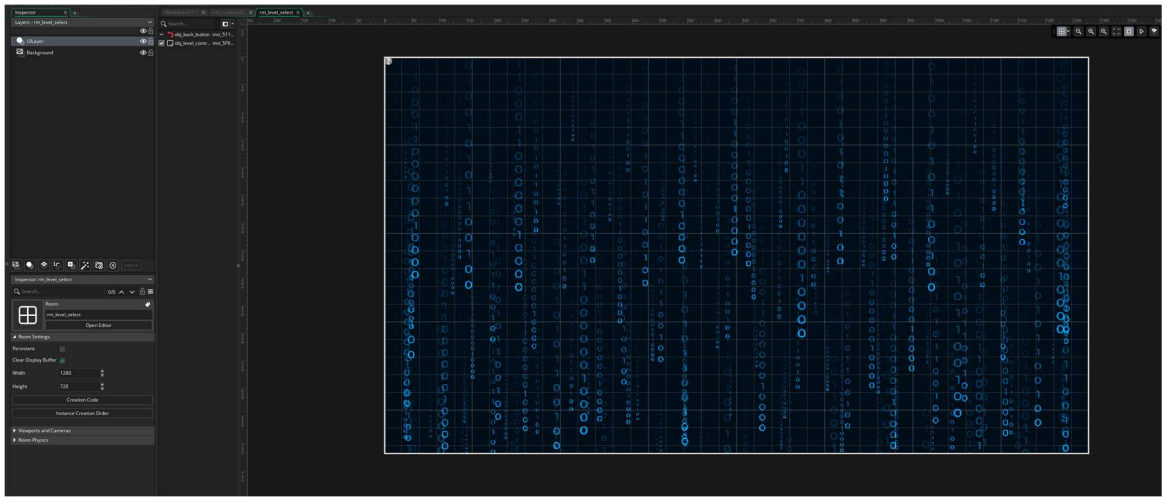
This appendix provides documentation of Alan's Coding Journey's source code and development environment. The following sections detail the core components, development interfaces, and code structure that comprise the game. The Rooms and Sprites will have graphics; however, they do not contain any source code. Objects do not have graphics and only contain source code.

Rooms

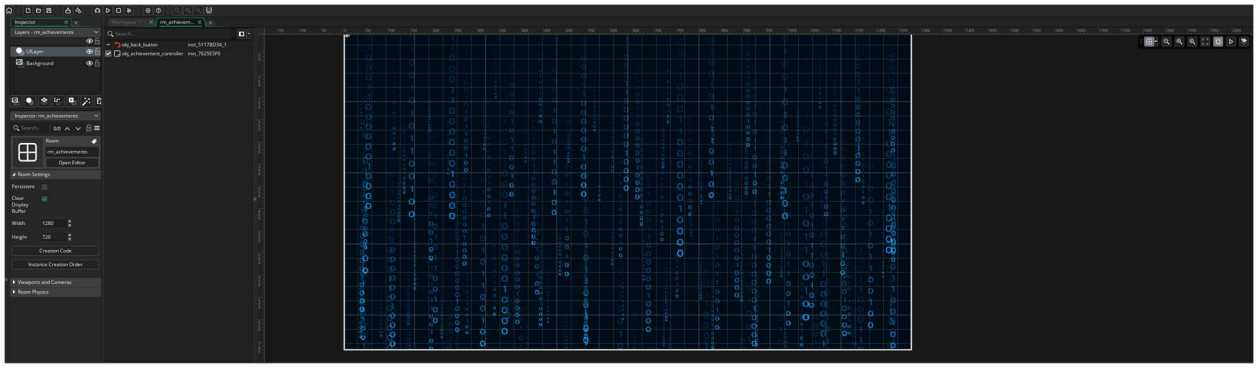
rm_start



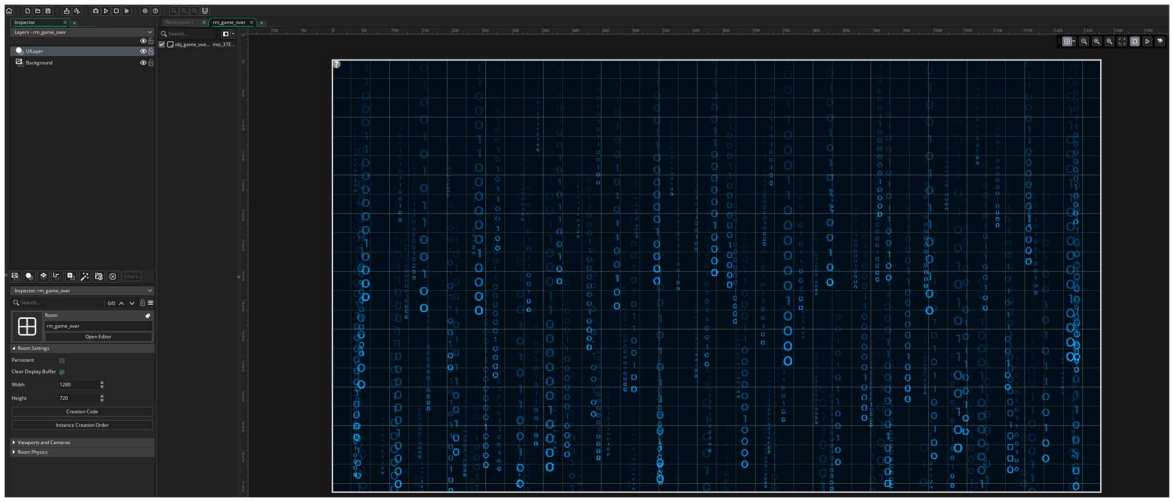
rm_level_select



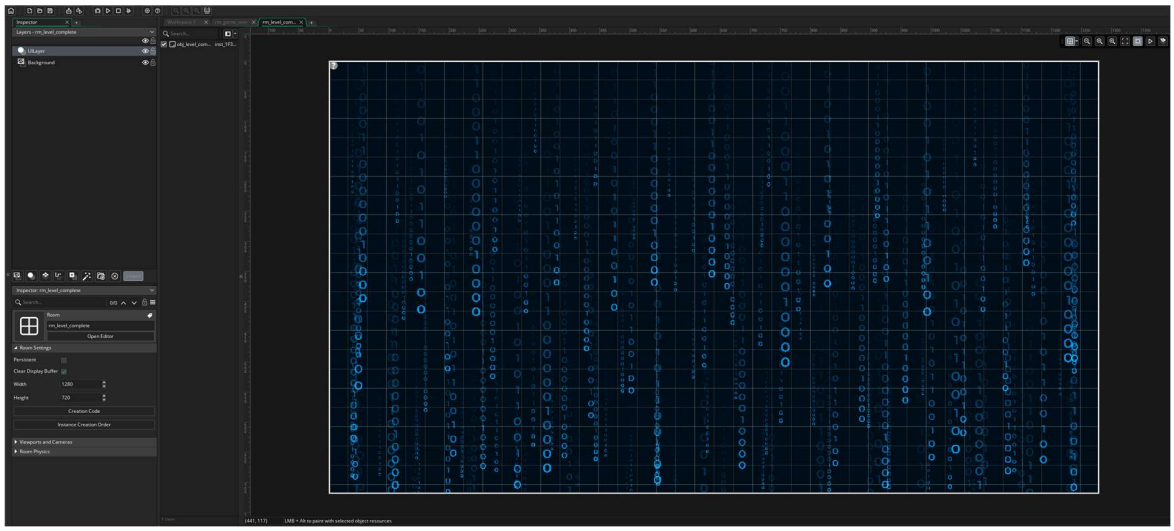
rm_game_over



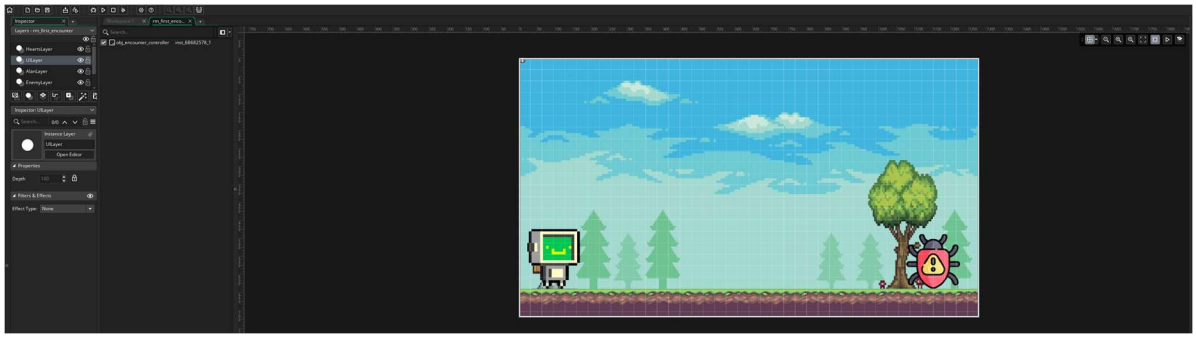
rm_game_over



rm_level_complete



rm_first_encounter



rm_first_learn

rm_second_encounter

rm_second_learn

rm_third_encounter

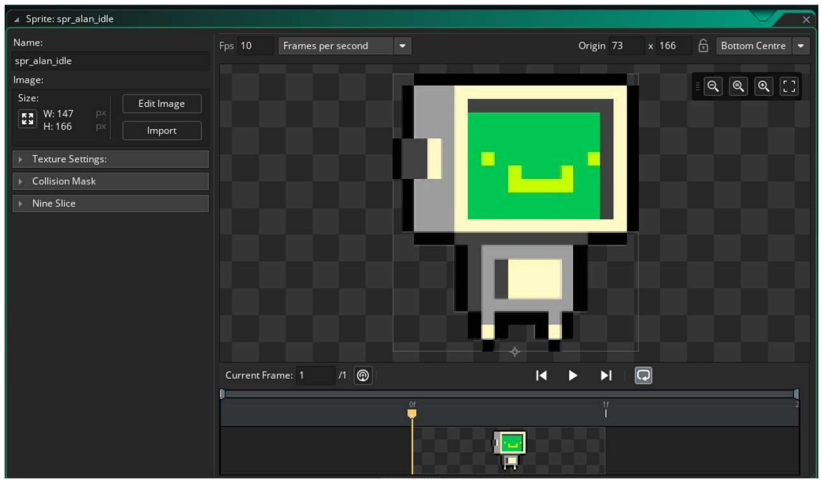
rm_third_learn

rm_fourth_encounter

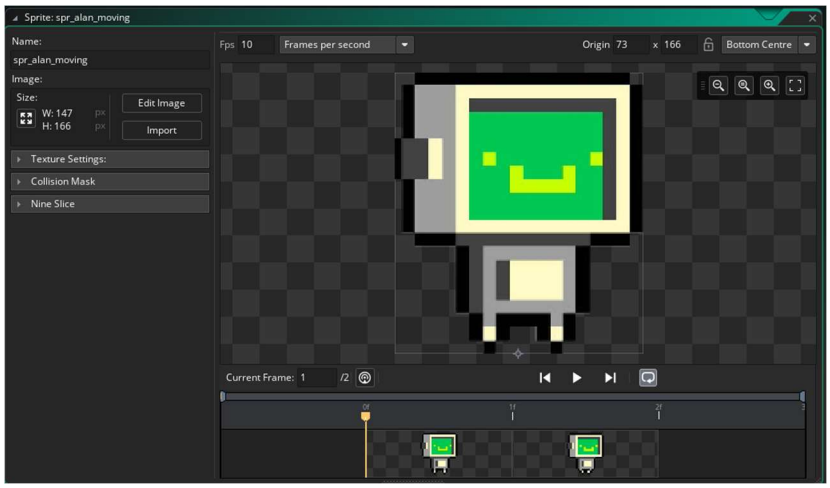
rm_fourth_learn

Sprites

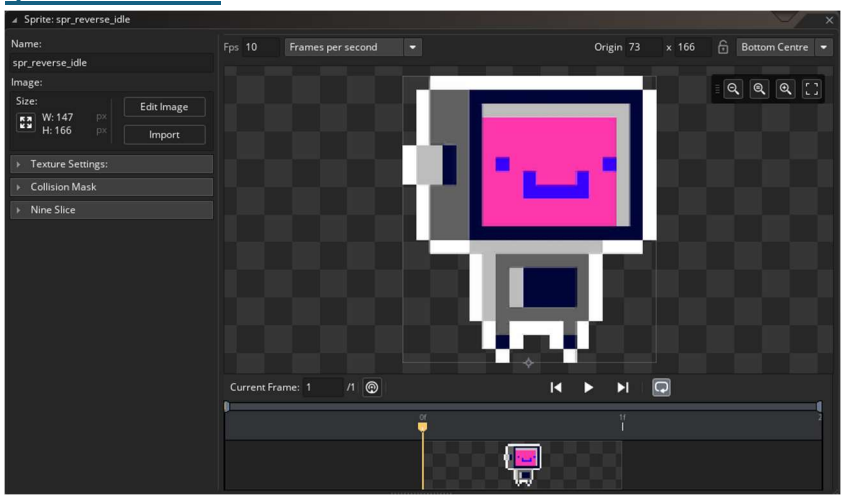
spr_alan_idle



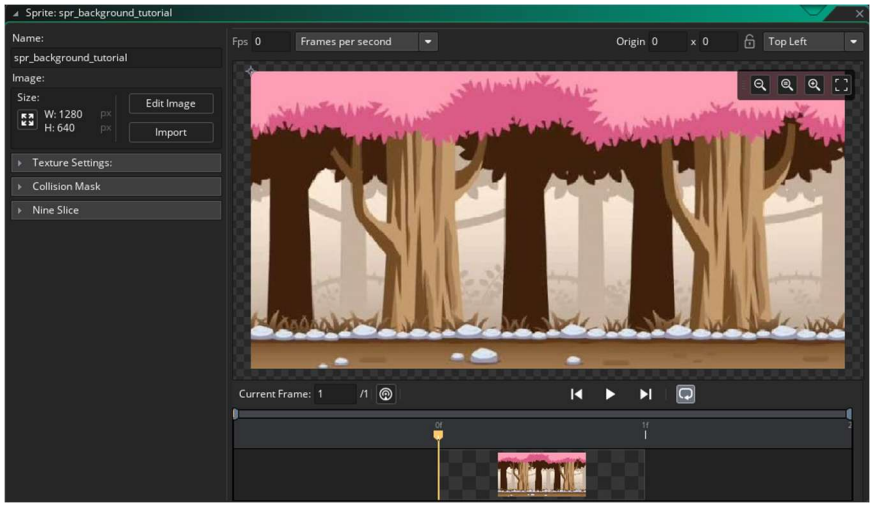
spr_alan_moving



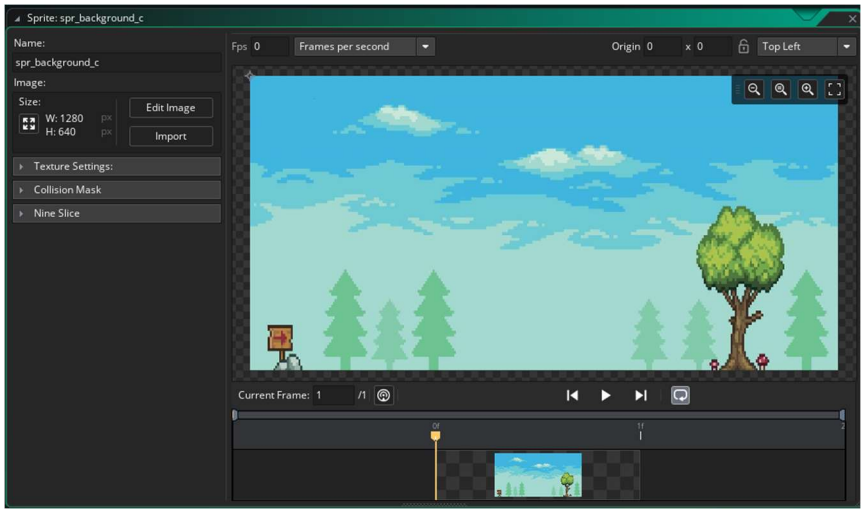
spr_reverse_idle



spr_background_tutorial

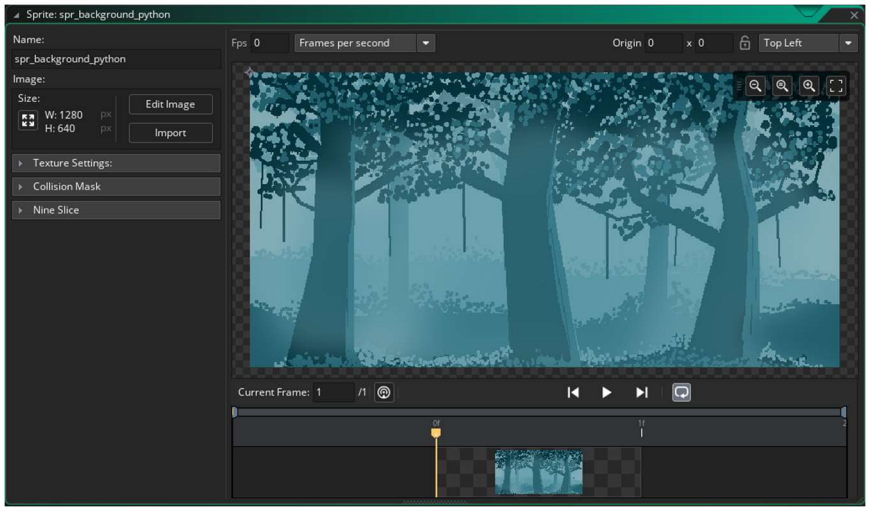


spr_background_first

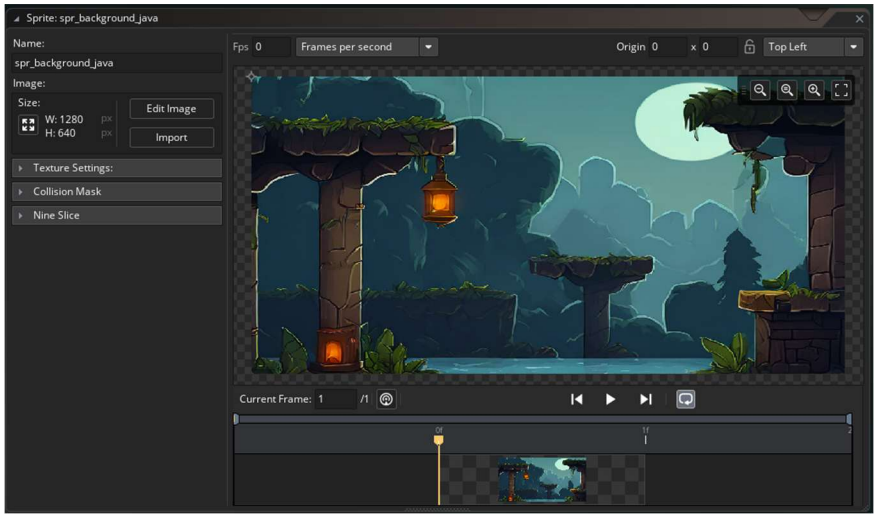


spr_background_second

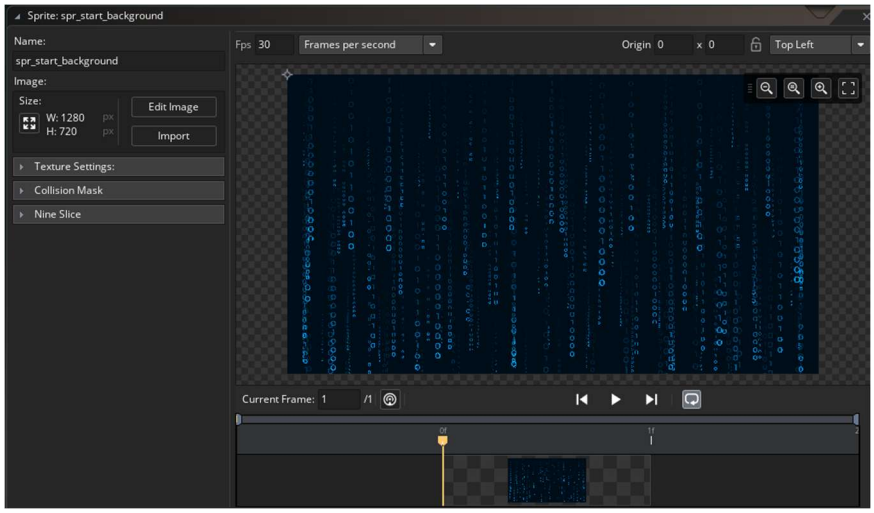
spr_background_third



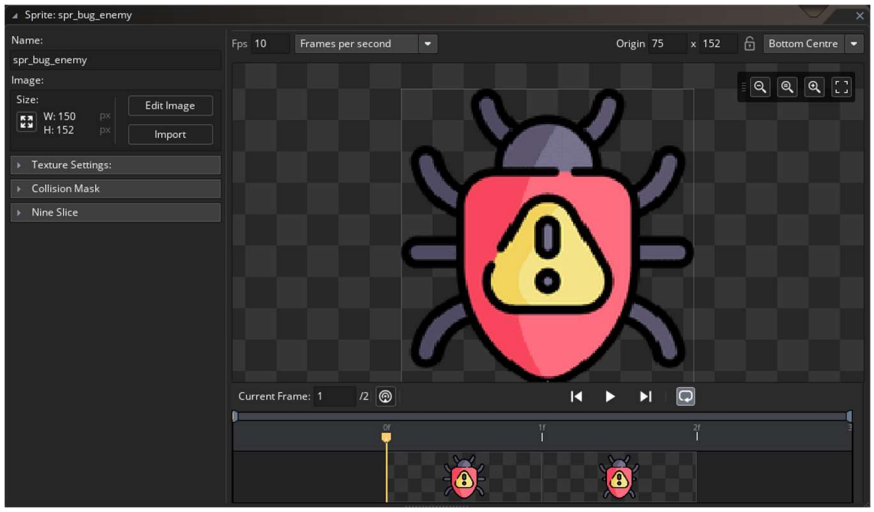
spr_background_fourth



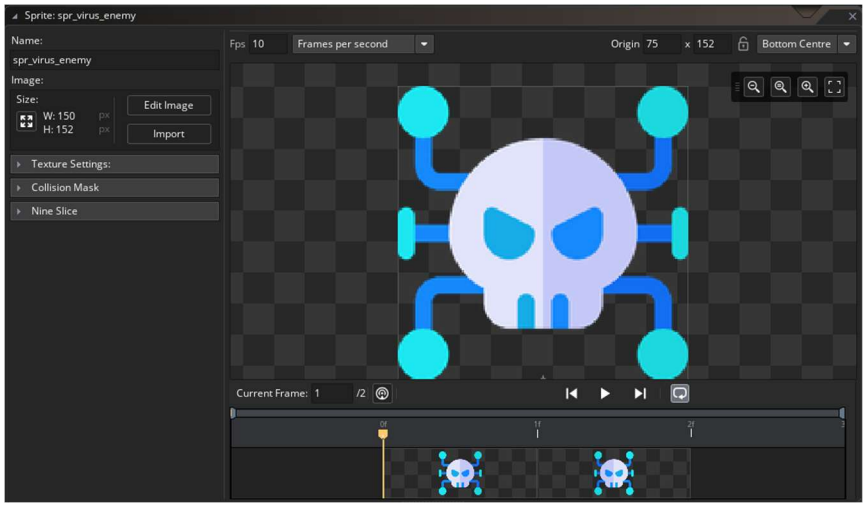
spr_start_background



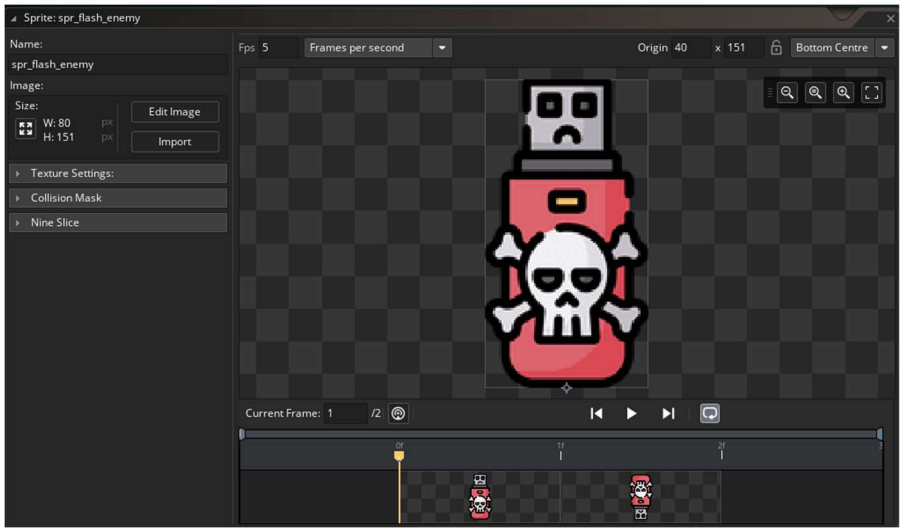
spr_bug_enemy



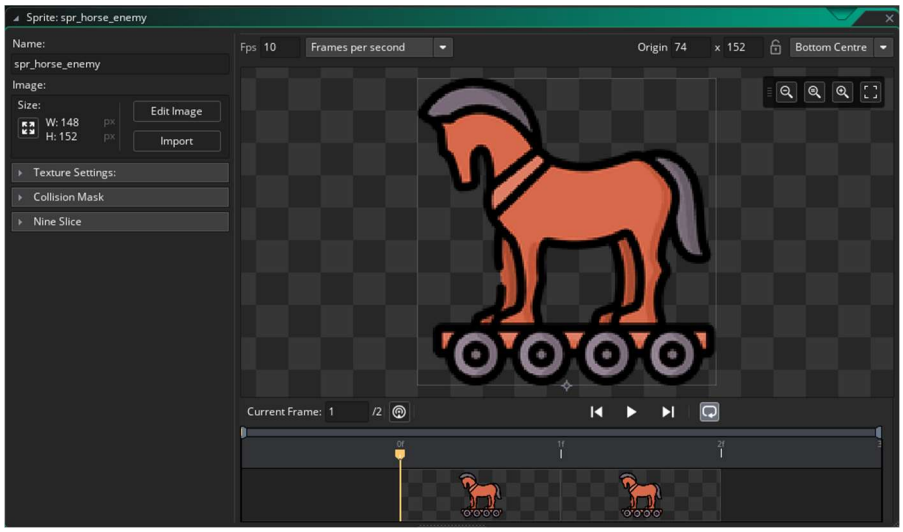
spr_virus_enemy



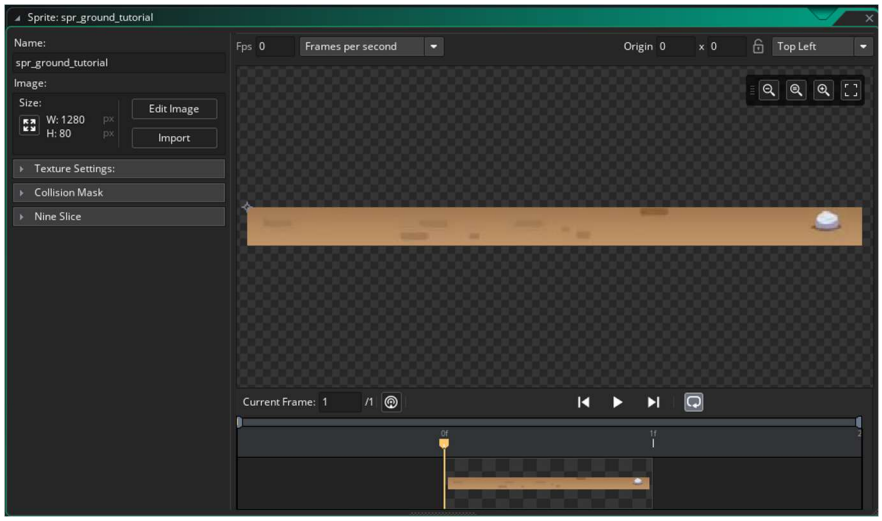
spr_flash_enemy



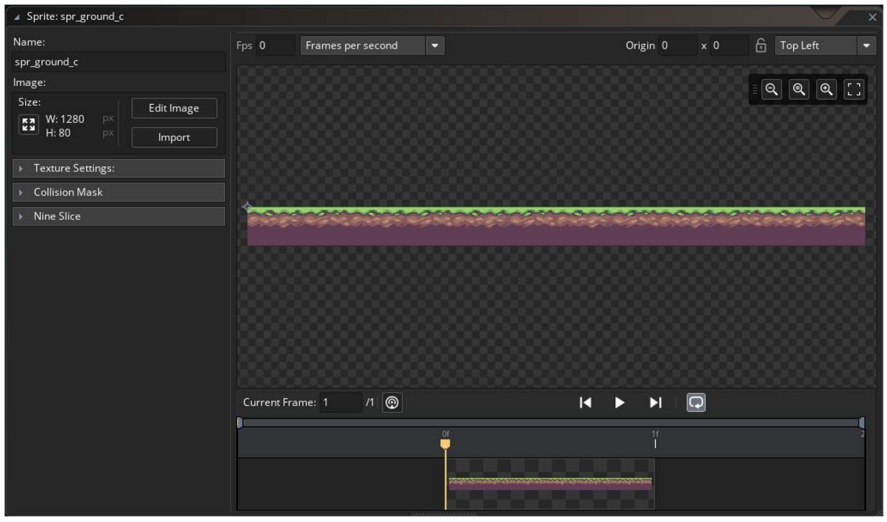
spr_horse_enemy



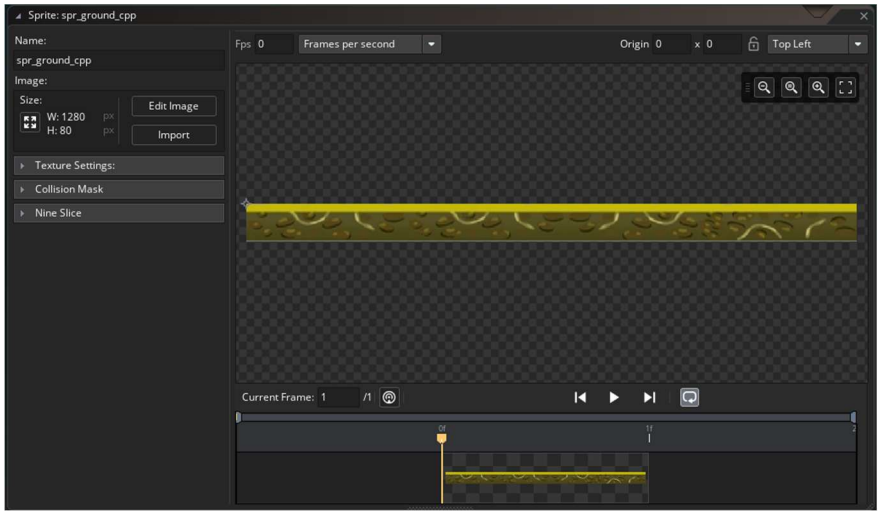
spr_ground_tutorial



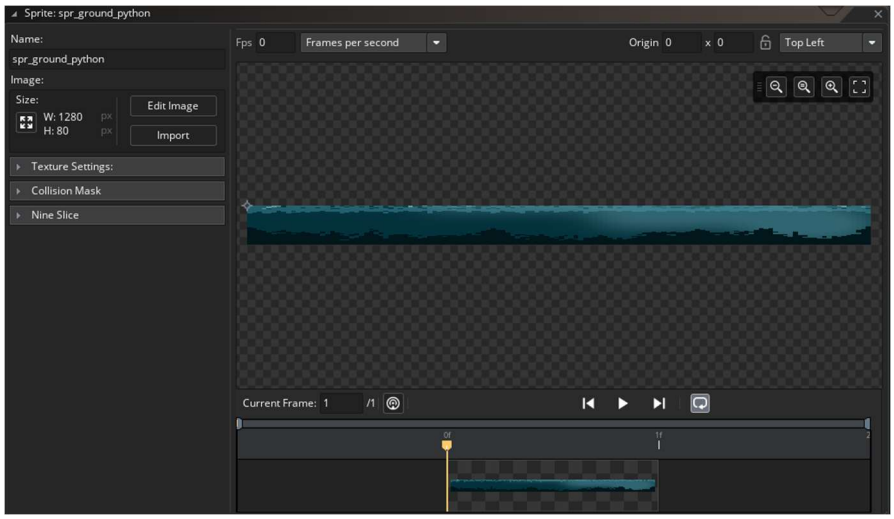
spr_ground_first



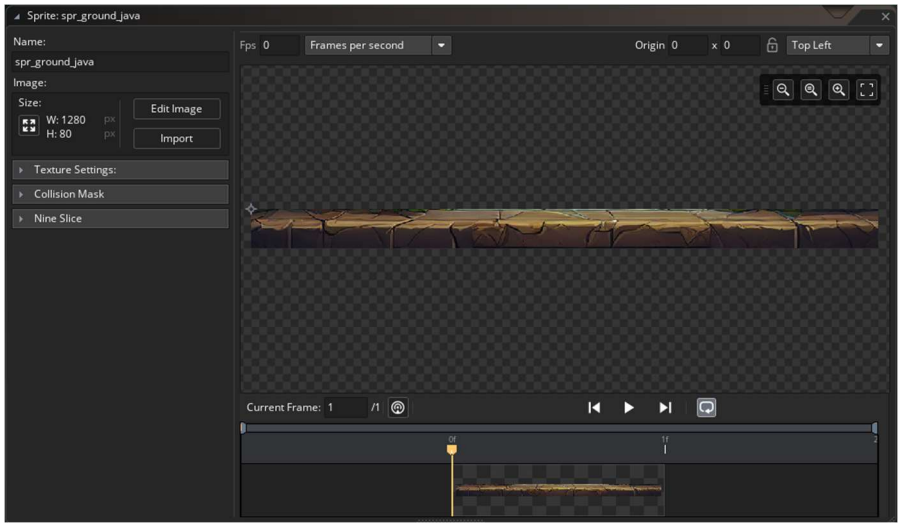
spr_ground_second



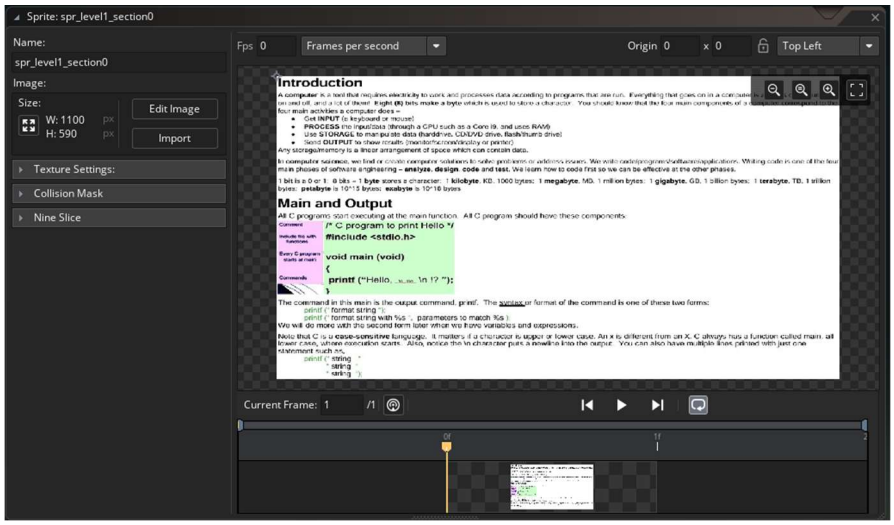
spr_ground_third



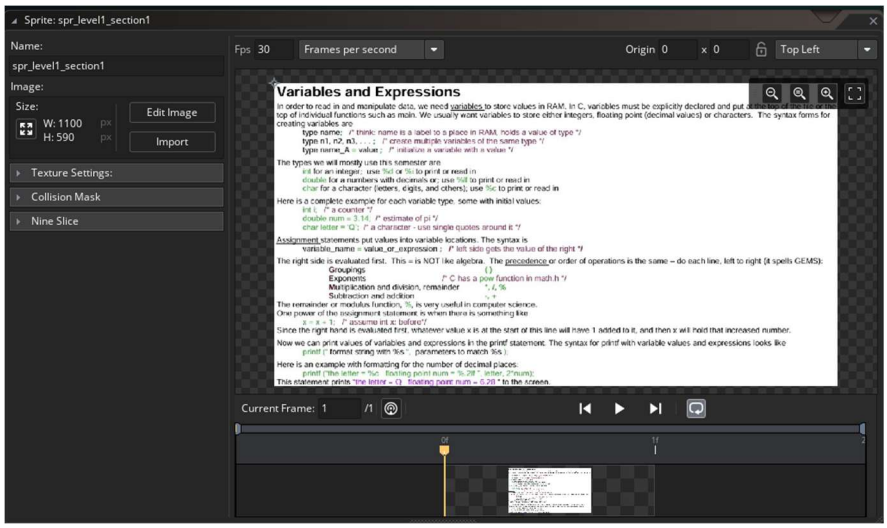
spr_ground_fourth



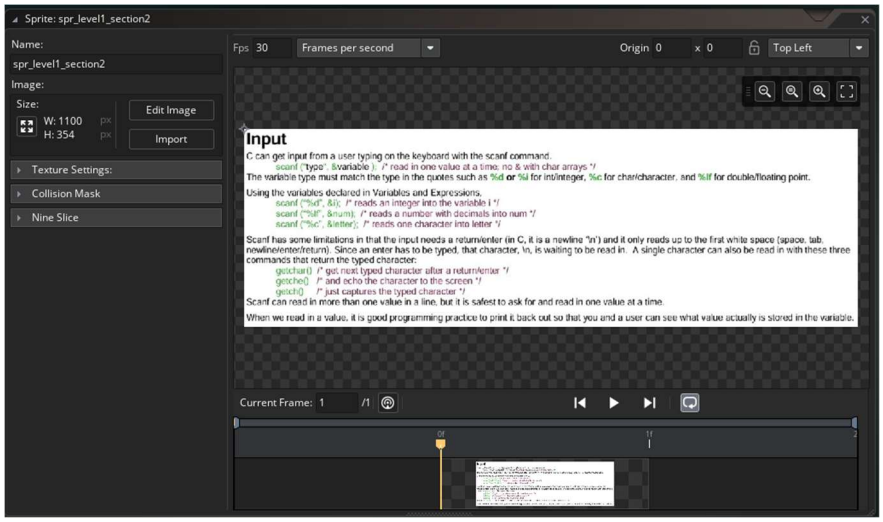
spr_level1_section0



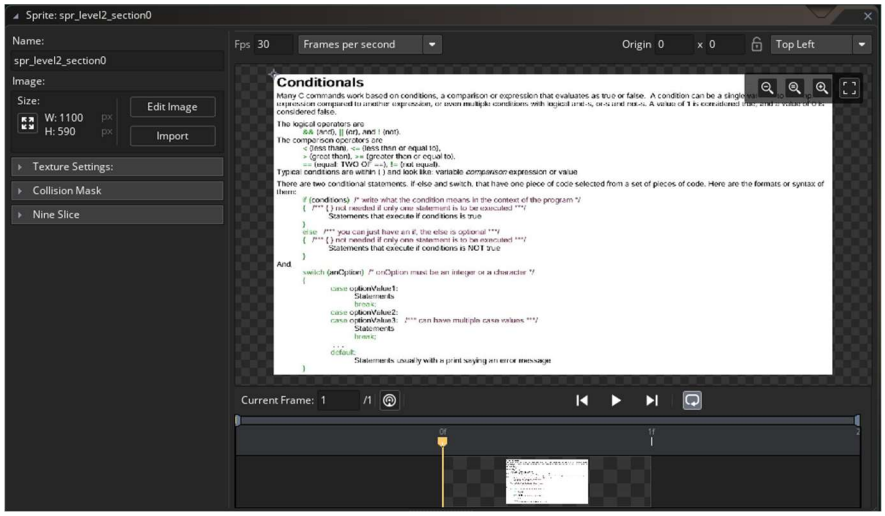
spr_level1_section1



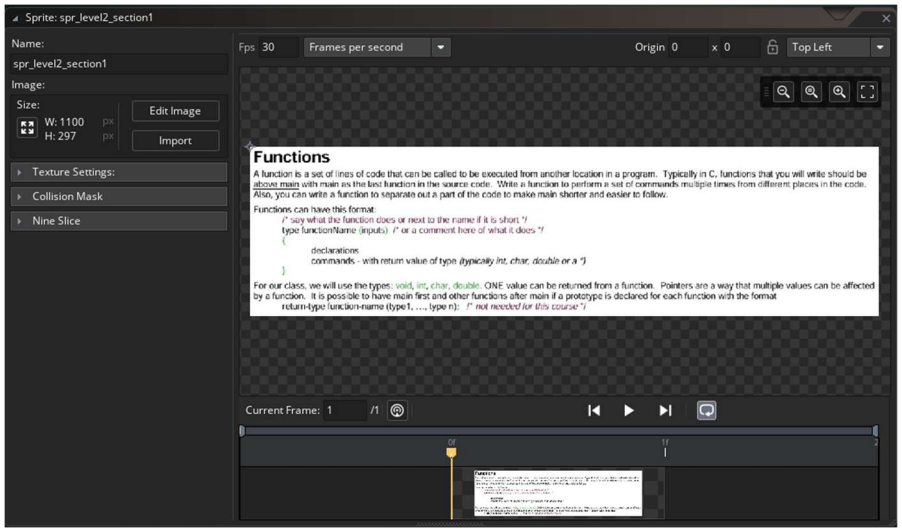
spr_level1_section2



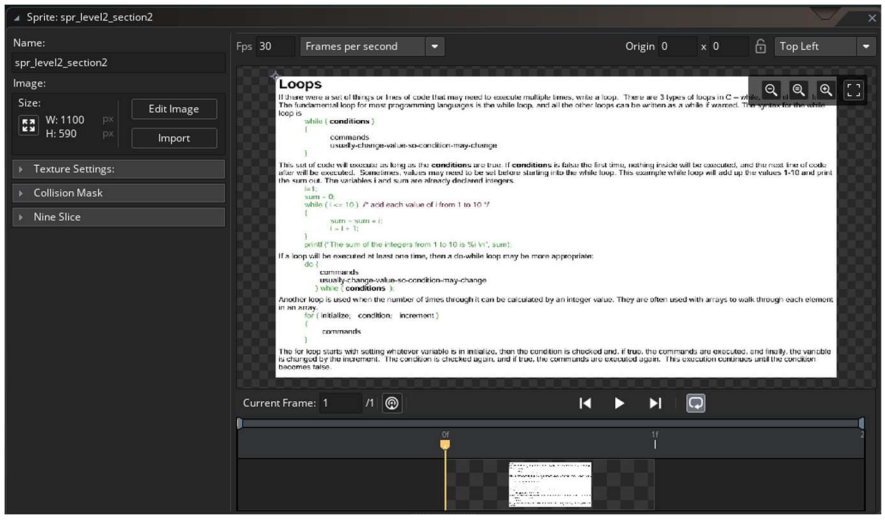
spr_level2_section0



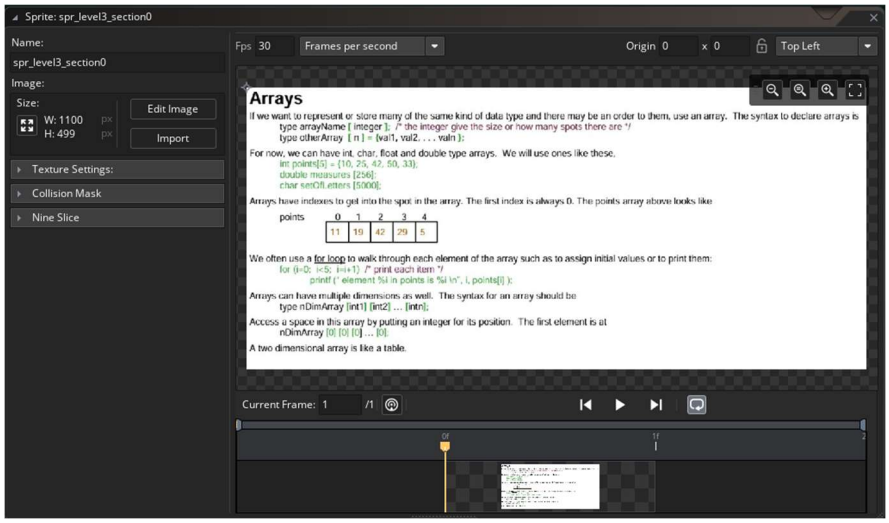
spr_level2_section1



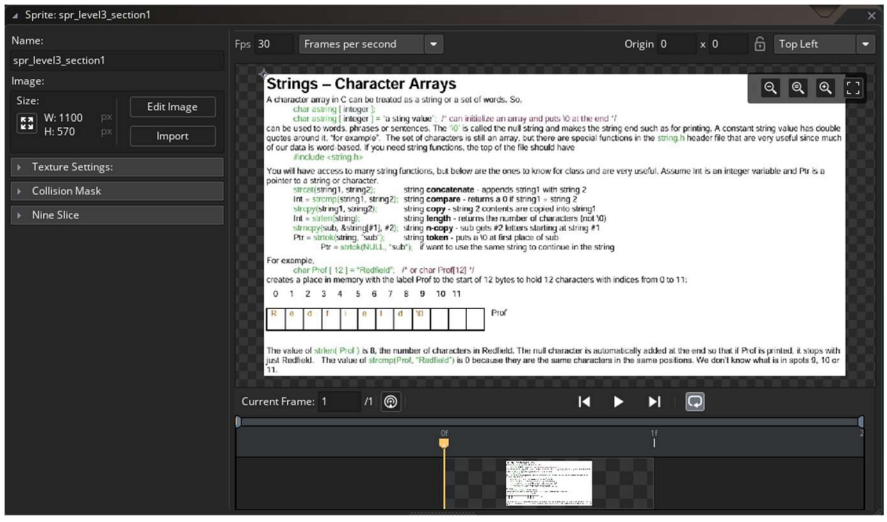
spr_level2_section2



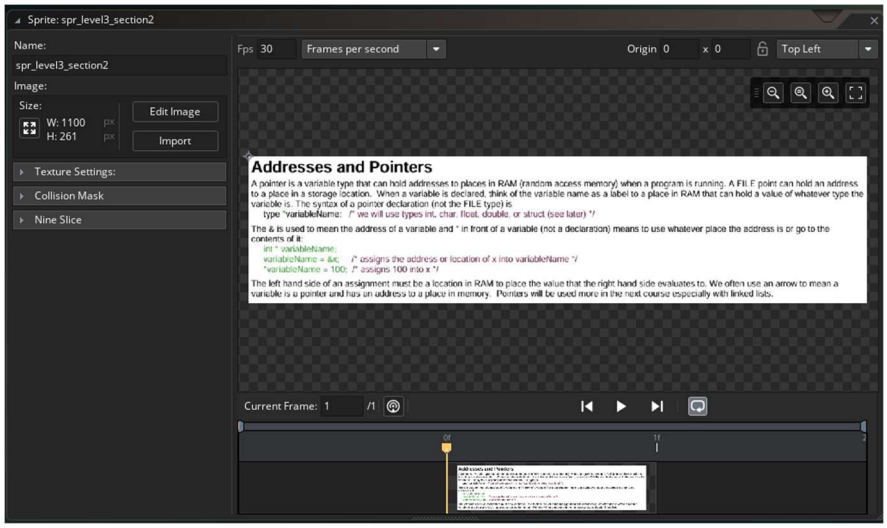
spr_level3_section0



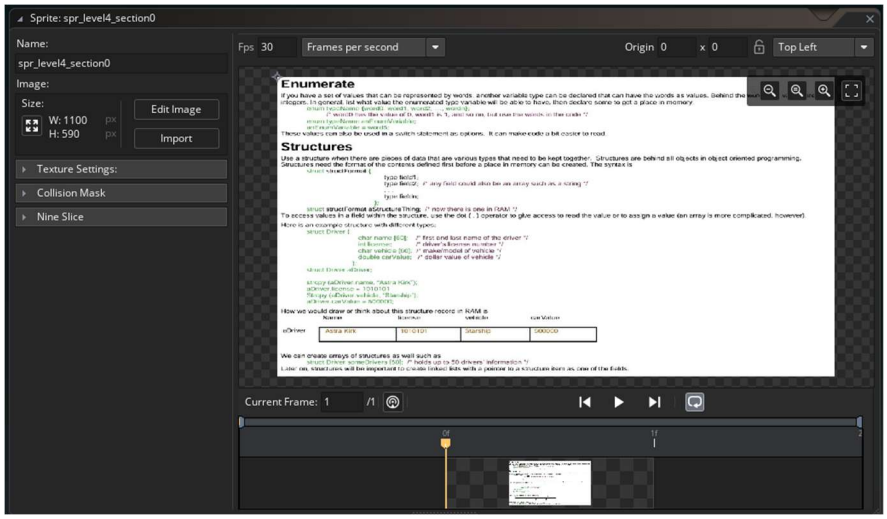
spr_level3_section1



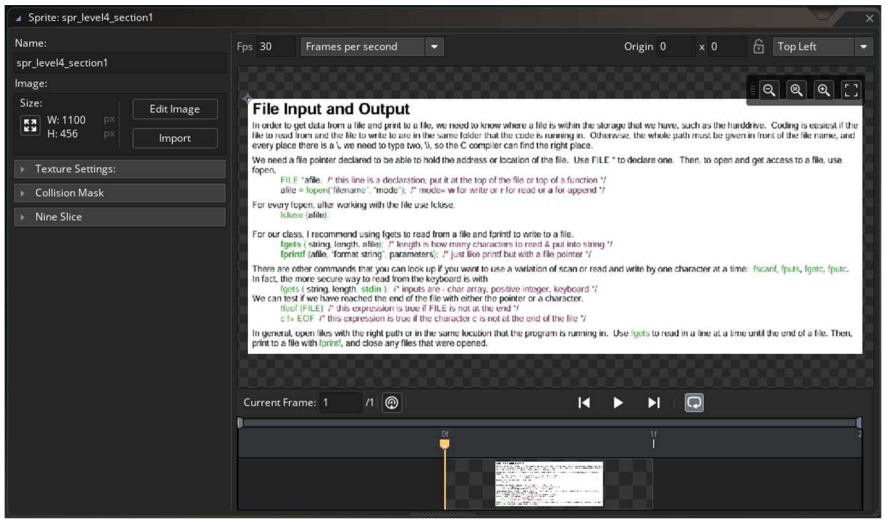
spr_level3_section2



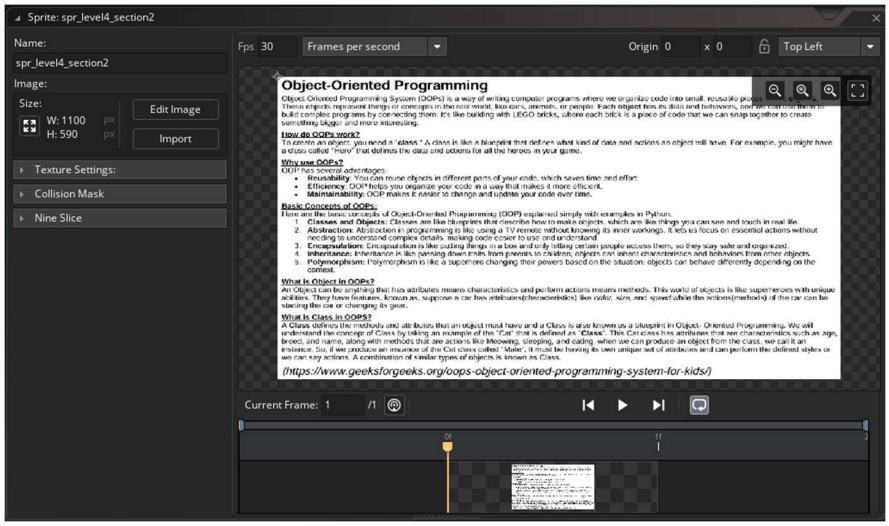
spr_level4_section0



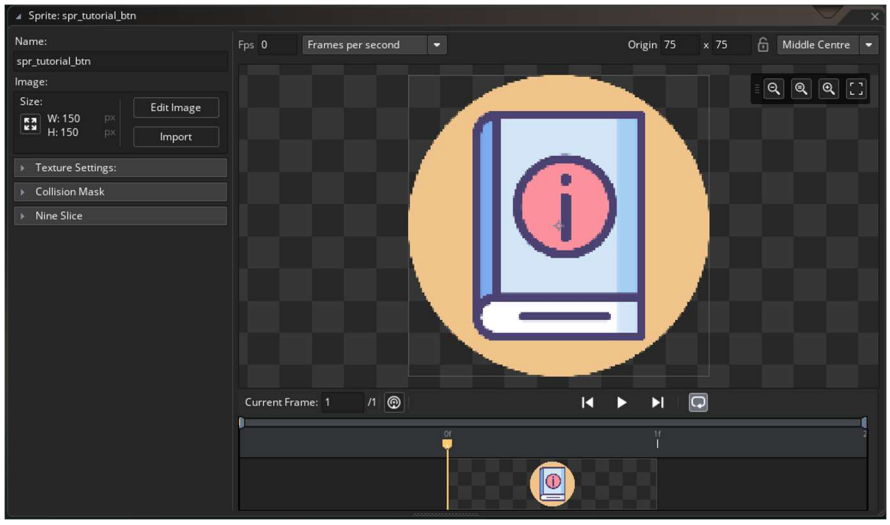
spr_level4_section1



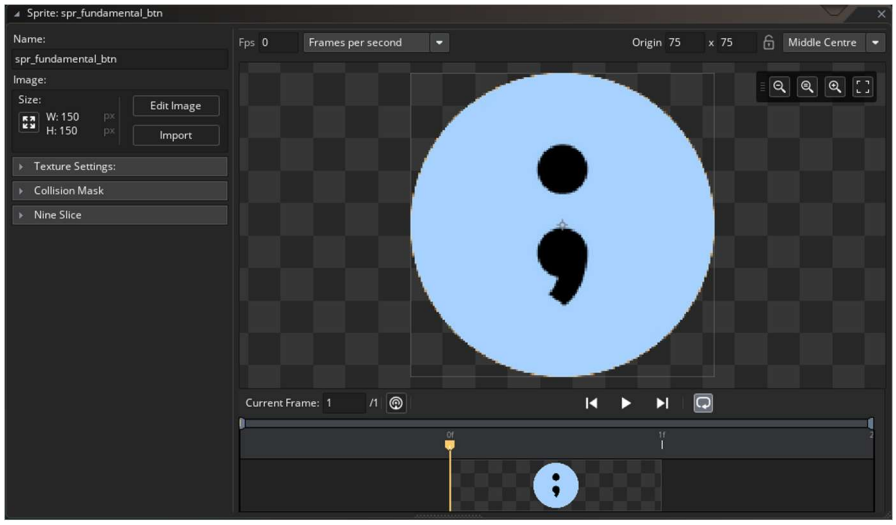
spr_level4_section2



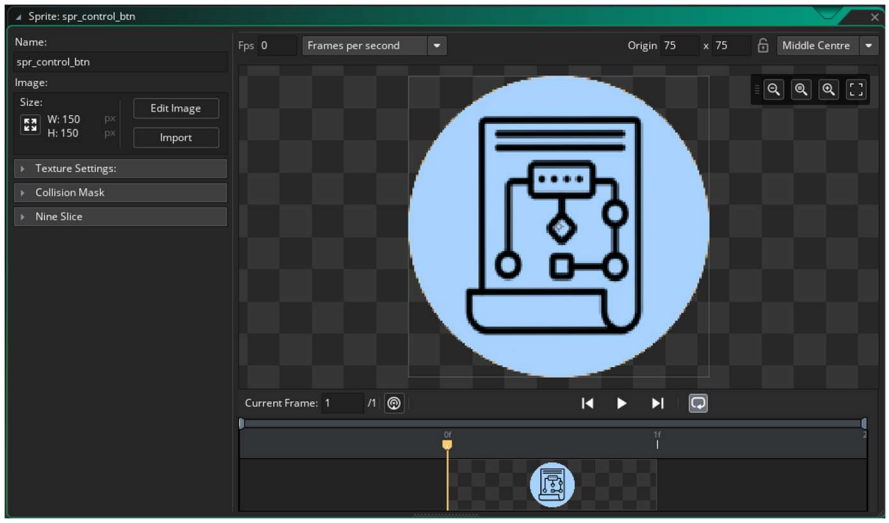
spr_tutorial_btn



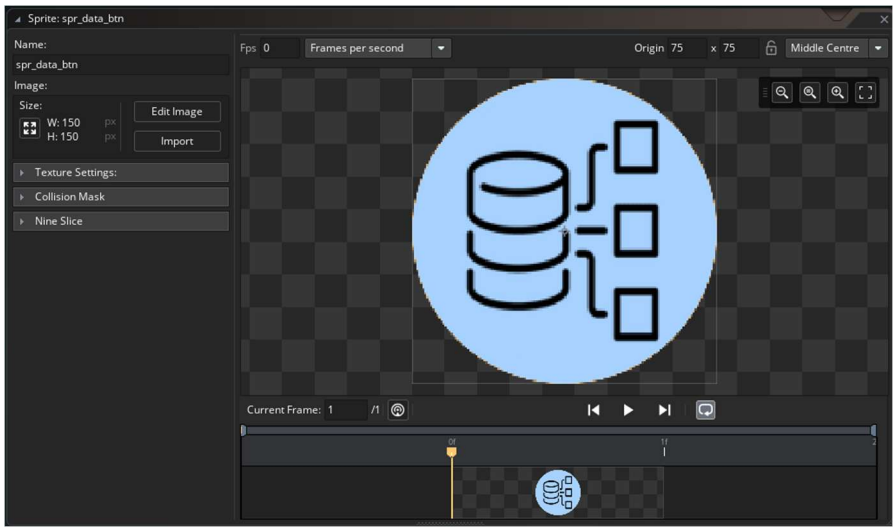
spr_fundamental_btn



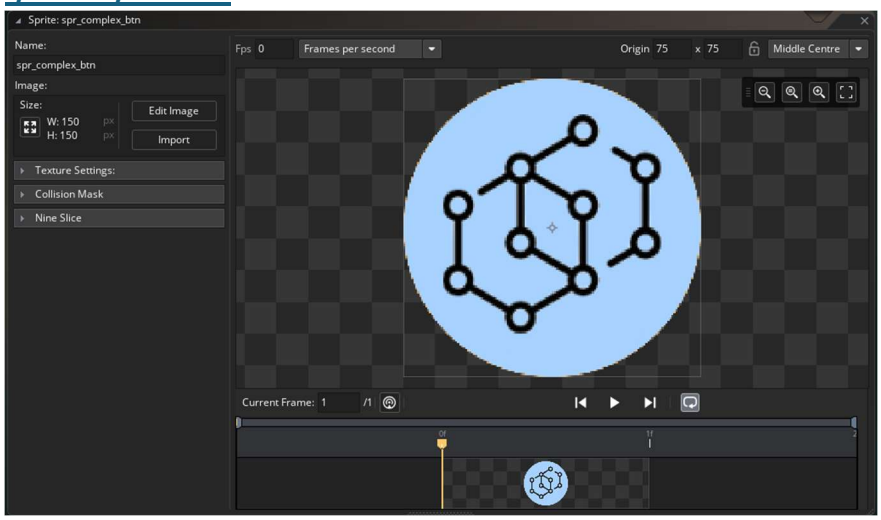
spr_control_btn



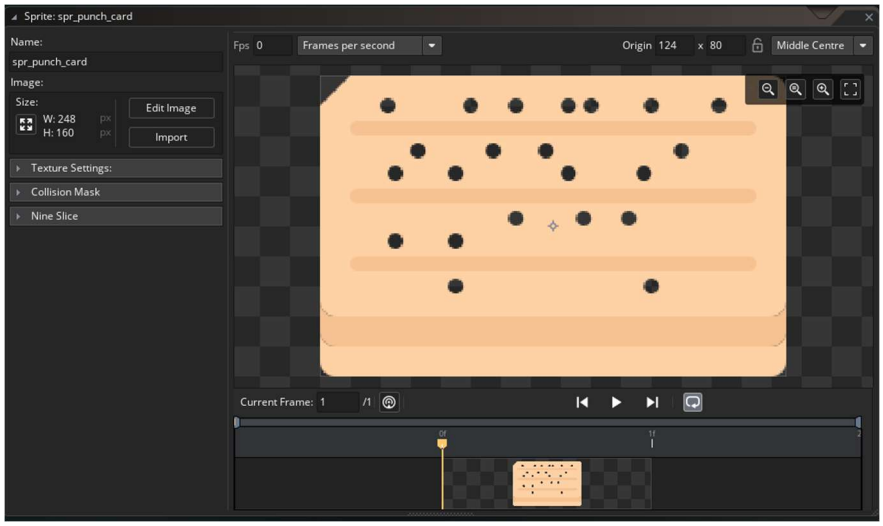
spr_data_btn



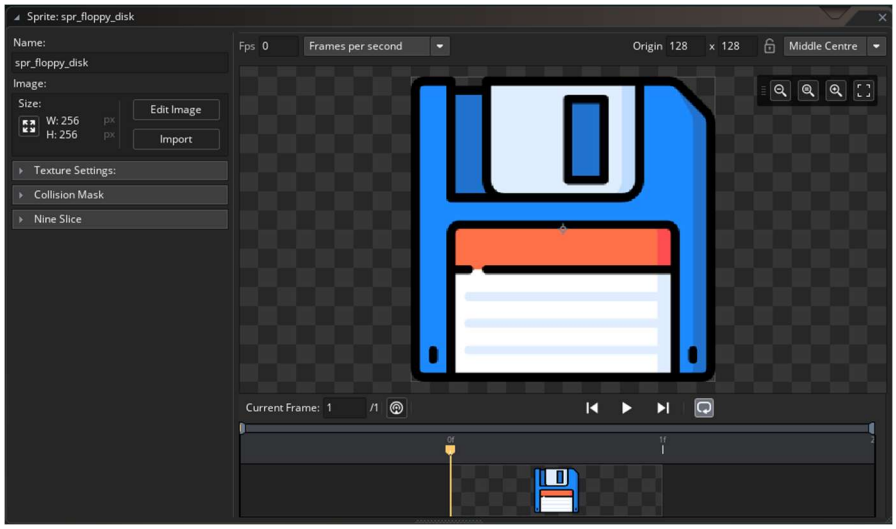
spr_complex_btn



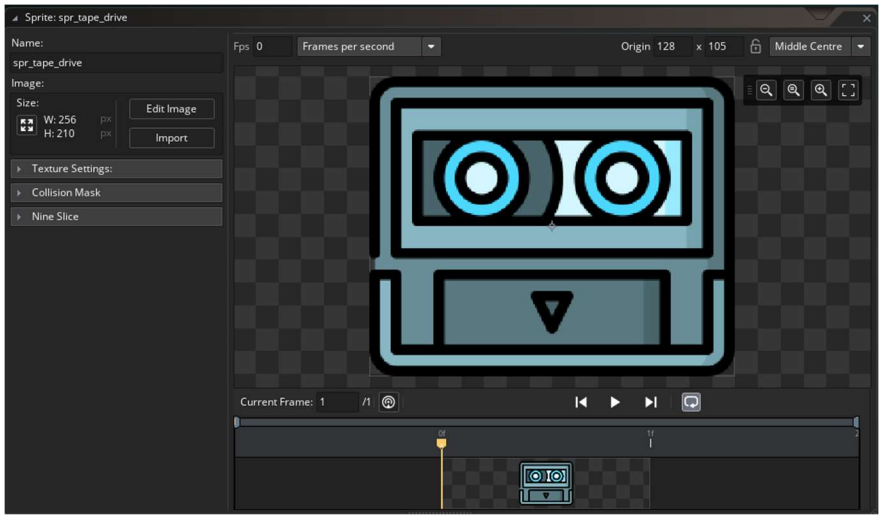
spr_punch_card



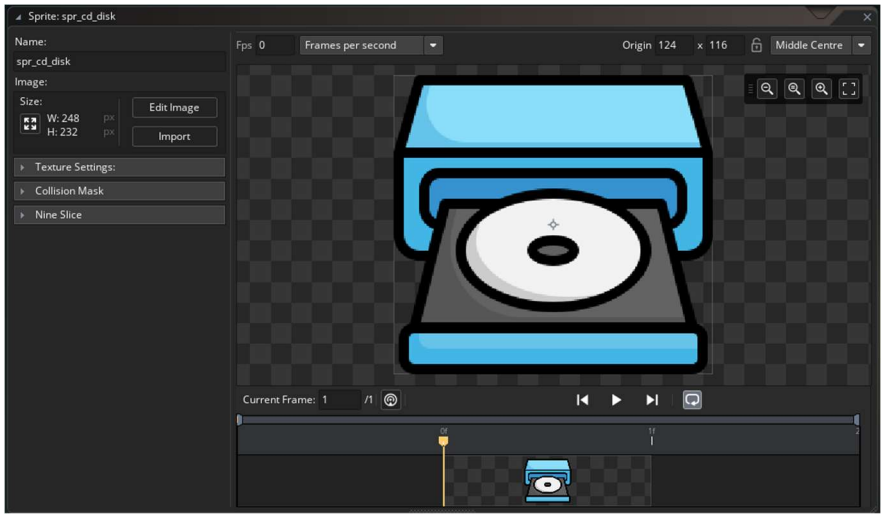
spr_floppy_disk



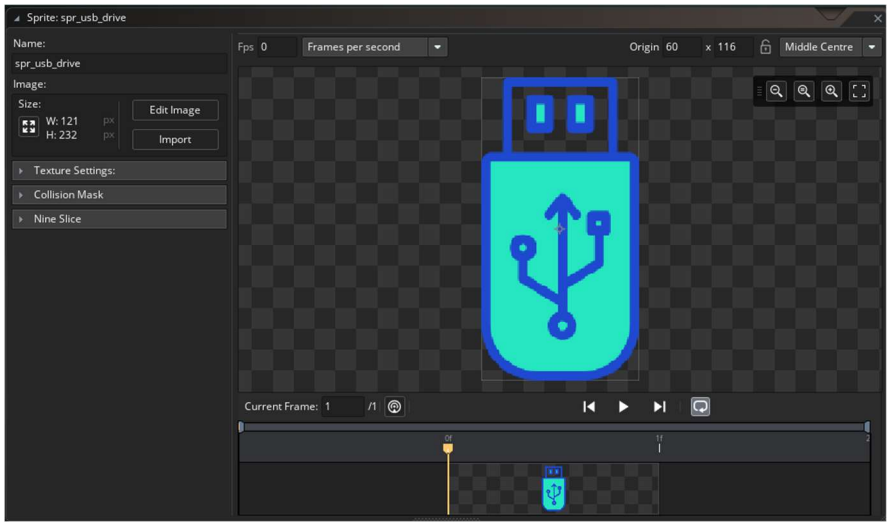
spr_tape_drive



spr_cd_disk

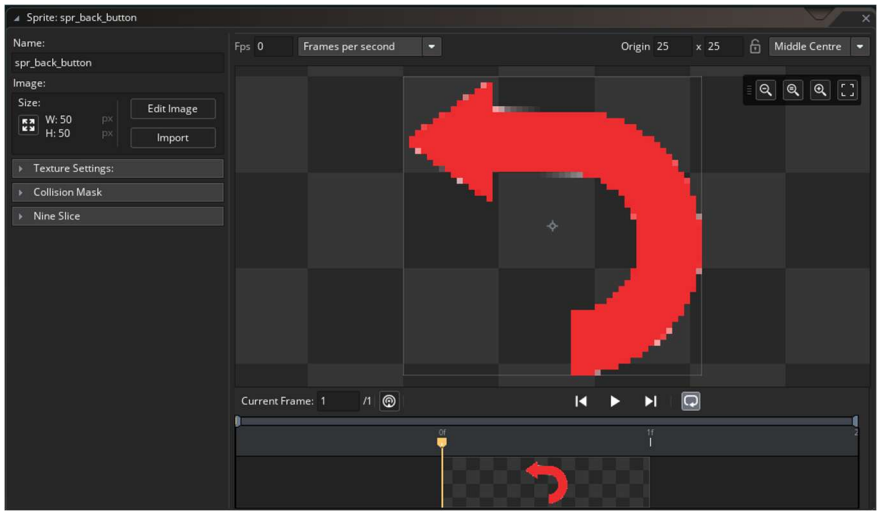


spr_usb_drive

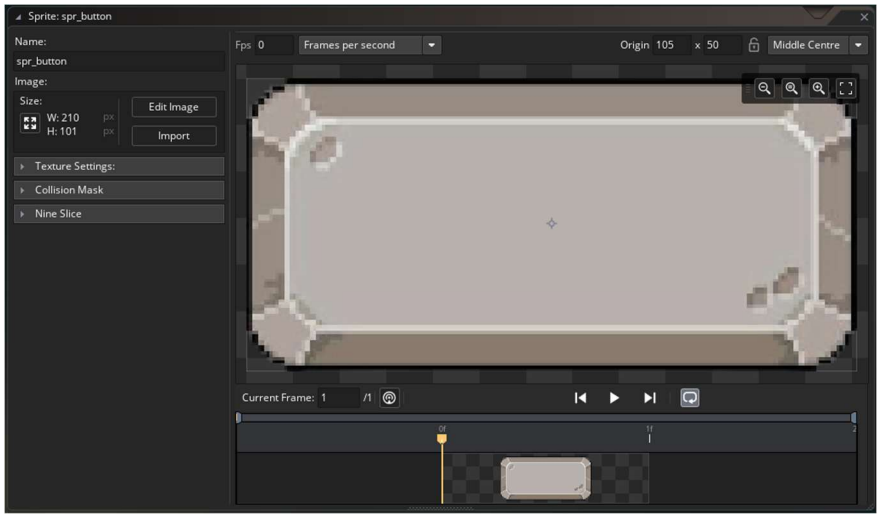


spr_wizard

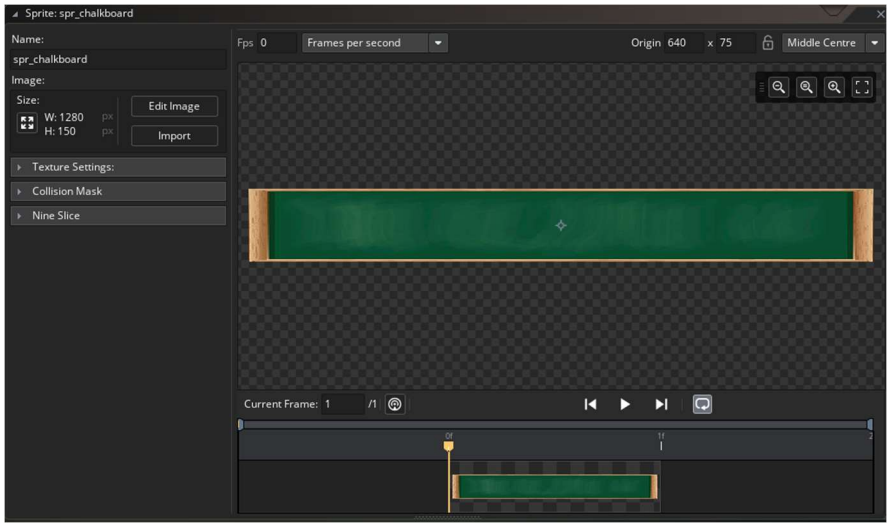
spr_back_button



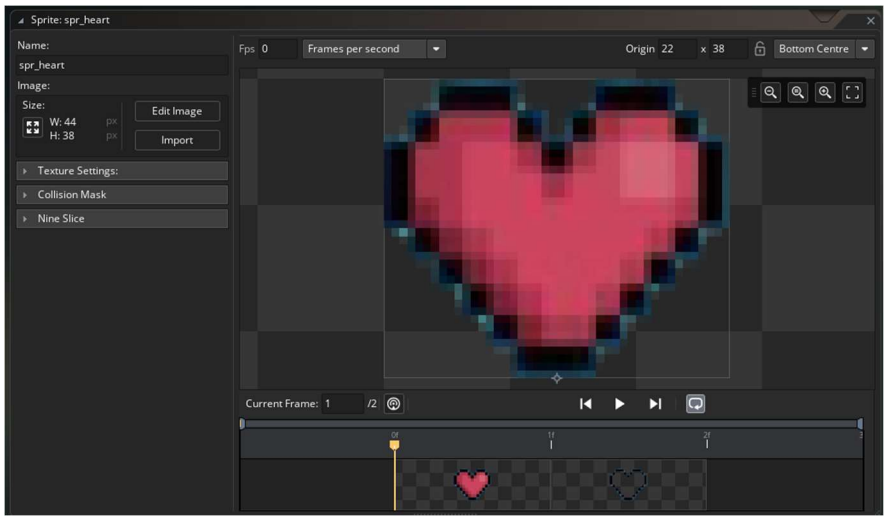
spr_button



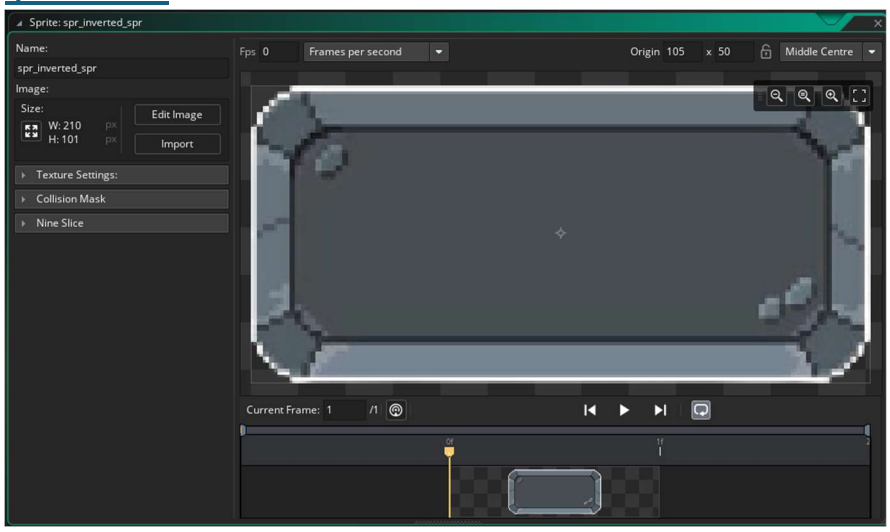
spr_chalkboard



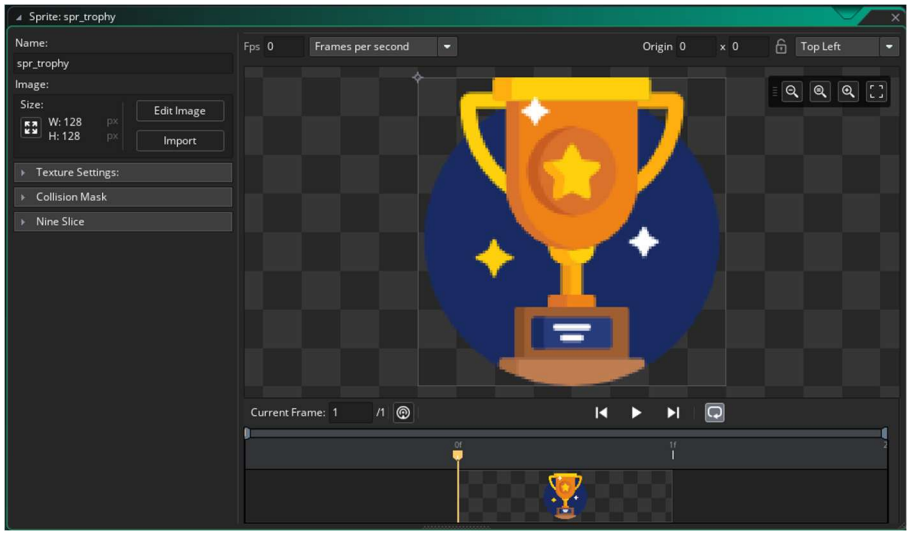
spr_heart



spr_inverted



spr_trophy


Sprite: spr_trophy
Name:
spr_trophy
Image:
Size:
W: 128
H: 128
px
px
Edit Image
Import
Texture Settings:
Collision Mask
Nine Slice
Fps 0
Frames per second
Origin 0 x 0
Top Left
Current Frame: 1 /1

Objects (Source Code)

obj_achievement_controller

Create.gml

```
// obj_level_achievement_controller Create Event
// Create back button
backButton = instance_create_layer(64, 64, "UILayer", obj_back_button);
backButton.backButton_type = "level";

//Create Top Left Trophy
firstTrophy = instance_create_layer(260, 250, "UILayer", obj_achievement_trophy);
firstTrophy.level_type = 0;
firstTrophy.sprite_index = spr_punch_card;
firstTrophy.unlocked = (global.highest_level - 1) >= 0;

//Create Top Middle Level Trophy
secondTrophy = instance_create_layer(640, 250, "UILayer", obj_achievement_trophy);
secondTrophy.level_type = 1;
secondTrophy.sprite_index = spr_tape_drive;
secondTrophy.unlocked = ((global.highest_level - 1) >= 1); //true;

//Create Top Right Level Trophy
thirdTrophy = instance_create_layer(1020, 250, "UILayer", obj_achievement_trophy);
thirdTrophy.level_type = 2;
thirdTrophy.sprite_index = spr_floppy_disk;
thirdTrophy.unlocked = ((global.highest_level - 1) >= 2); //true;

//Create Bottom Left Trophy
fourthTrophy = instance_create_layer(260, 550, "UILayer", obj_achievement_trophy);
fourthTrophy.level_type = 3;
fourthTrophy.sprite_index = spr_cd_disk;
fourthTrophy.unlocked = ((global.highest_level - 1) >= 3); //true;

//Create Bottom Middle Level Trophy
fifthTrophy = instance_create_layer(640, 550, "UILayer", obj_achievement_trophy);
fifthTrophy.level_type = 4;
fifthTrophy.sprite_index = spr_usb_drive;
fifthTrophy.unlocked = ((global.highest_level >= 4) && (global.game_complete ==
true)); //true;

//Create Bottom Right Level Trophy
sixthTrophy = instance_create_layer(1020, 550, "UILayer", obj_achievement_trophy);
sixthTrophy.level_type = 5;
sixthTrophy.sprite_index = spr_wizard;
sixthTrophy.unlocked = ((global.highest_level >= 4) && (global.game_complete ==
true)); //true;
```

DrawGUI.gml

```
// Title
draw_set_font(fnt_title);
```

```
draw_set_halign(fa_center);
draw_text(room_width/2, 100, "Trophies");

//Reset
draw_set_font(fnt_button);
```

obj_achievement_trophy

Create.gml

```
// obj_level_button Create Event
level_type = -1;  // Will be set according to global.current_level values
show_debug_message("Trophy level: " + string(level_type));
show_debug_message("Highest Trophy: " + string(global.highest_level));
unlocked = (level_type <= global.highest_level);
show_debug_message("Trophy unlocked: " + string(unlocked));
```

Draw.gml

```
// Draw Event
draw_self();

draw_set_color(c_white);
draw_set_font(fnt_select);

switch(level_type)
{
	case 0:
		draw_text(x, y + (sprite_height/2) + 35, "Complete Tutorial");
		break;
	case 1:
		draw_text(x, y + (sprite_height/2) + 35, "Complete Fundamentals Level");
		break;
	case 2:
		draw_text(x, y + (sprite_height/2) + 35, "Complete Control Flows
Level");
		break;
	case 3:
		draw_text(x, y + (sprite_height/2) + 35, "Complete Data Structures
Level");
		break;
	case 4:
		draw_text(x, y + (sprite_height/2) + 35, "Complete Complex Concepts
Level");
		break;
	case 5:
		draw_text(x, y + (sprite_height/2) + 35, "Complete All Levels");
		break;
}

if (!unlocked)
{
	draw_set_alpha(1);
	draw_set_color(c_gray);
```

```
      image_speed = 0;
    draw_sprite_ext(sprite_index, 0, x, y, image_xscale, image_yscale, 0, c_black,
1);

      draw_set_alpha(1);
    draw_text(x, y + (sprite_height/2) + 10 , "??????????");

      // Reset alpha and color after drawing this button
      draw_set_color(c_white);

}
else
{
      draw_set_color(c_yellow);
      draw_set_font(fnt_select);
      image_speed = 1;
      draw_sprite_ext(sprite_index, 0, x, y, image_xscale, image_yscale, 0, c_white,
1);


      switch(level_type)
      {
            case 0:
                  draw_text(x, y + (sprite_height/2) + 10, "Punch Card Pioneer");
                  break;
            case 1:
                  draw_text(x, y + (sprite_height/2) + 10, "Tape Master");
                  break;
            case 2:
                  draw_text(x, y + (sprite_height/2) + 10, "Floppy Expert");
                  break;
            case 3:
                  draw_text(x, y + (sprite_height/2) + 10, "CD Champion");
                  break;
            case 4:
                  draw_text(x, y + (sprite_height/2) + 10, "USB Virtuoso");
                  break;
            case 5:
                  draw_text(x, y + (sprite_height/2) + 10, "Storage Wizard");
                  break;
      }

      draw_set_color(c_white);
```

obj_trophy_button

Create.gml

```
// obj_trophy_button Create Event
depth = -10001;
hovering = false;
clicked = false;move_speed = 5;
```

DrawGUI.gml

```
draw_self();
```

LeftPressed.gml

```
// Left Pressed Event
image_alpha = 1;
clicked = true;
```

LeftReleased.gml

```
image_alpha = 0.5;

if (hovering && clicked)
{
	clicked = false;
	room_goto(rm_achievements);
}
else
{
	clicked = false;
}
```

MouseEnter.gml

```
// Mouse Enter Event
image_alpha = 0.5;
hovering = true;
```

MouseLeave.gml

```
// Mouse Leave Event
image_alpha = 1;
hovering = false;
clicked = false;
```

obj_alan

Create.gml

```
hspeed = 0;
vspeed = 0;
grav = 0.5;
jump_speed = -12;
move_speed = 5;

// Animation and direction control
facing = 1;           // 1 for right, -1 for left
sprite_idle = spr_alan_idle;    // Your idle sprite
sprite_run = spr_alan_moving; // Your running animation sprite
image_speed = 0;      // Start with animation stopped

// Ensure proper placement
depth = -100; // Makes sure Alan appears in front

// Fix size consistency
```

```
image_xscale = 1; // Or whatever scale you want
image_yscale = 1; // Keep it the same as xscale


invincible = false;
invincible_time = 120;  // How long invincibility lasts (120 frames = 2 seconds)
invincible_timer = 0;
flash = false;  // For visual feedback when invincible
```

Step.gml

```
if (global.can_move)
{
	// Get movement input
	var move = keyboard_check(vk_right) - keyboard_check(vk_left);
	hspeed = move * move_speed;

	// Gravity and jumping
	vspeed += grav;

	// Horizontal boundary checking
	var sprite_half_width = abs(sprite_width/2); // Use absolute value to handle
flipping

	if (x + hspeed < sprite_half_width)
	{
		x = sprite_half_width;
		hspeed = 0;
	}

	if(room == rm_tutorial_learn && global.tutorial_lock = true)
	{
		 if (x + hspeed > room_width - sprite_half_width)
	  {
		x = room_width - sprite_half_width;
		hspeed = 0;
	  }
	}

	// Right boundary (only enforce in encounter room if enemy exists)
    if (room == rm_tutorial_encounter || room == rm_first_encounter || room ==
rm_second_encounter || room == rm_third_encounter || room == rm_fourth_encounter)
    {
        if (instance_exists(obj_enemy_parent))
        {
            if (x + hspeed > room_width - sprite_half_width)
            {
                x = room_width - sprite_half_width;
                hspeed = 0;
            }
        }
    }


	// Ground collision check for jumping
```

```
var on_ground = place_meeting(x, y + 1, obj_ground_parent);

// Jump when space is pressed and on ground
if (keyboard_check_pressed(vk_space) && on_ground)
{
	vspeed = jump_speed;
}

// Vertical collisions
if (place_meeting(x, y + vspeed, obj_ground_parent))
{
	while (!place_meeting(x, y + sign(vspeed), obj_ground_parent))
	{
		y += sign(vspeed);
	}

	vspeed = 0;
}

// Horizontal collisions
if (place_meeting(x + hspeed, y, obj_ground_parent))
{
	while (!place_meeting(x + sign(hspeed), y, obj_ground_parent))
	{
		x += sign(hspeed);
	}

	hspeed = 0;
}

if(!invincible)
{
	var enemy = instance_place(x, y, obj_enemy_parent);

	if (enemy != noone)
	{
		// If hit from side
	  if (y >= enemy.y - enemy.sprite_height/2)
		{
		// Reduce heart
		with(obj_heart_controller)
			{
		    current_hearts -= 1;
		    flash_hearts = true;
				flash_timer = 0;

				if(current_hearts <= 0)
				{
				   global.current_encounter = 1;
			current_hearts = max_hearts;  // Reset hearts
				   room_goto(rm_game_over);
		    }
		}

		// Push Alan away from enemy
```

```
                x = x + (200 * sign(x - enemy.x));  // Push in opposite direction
of enemy


                invincible = true;
                invincible_timer = invincible_time;

                 }
            }
      }

      // Handle invincibility timer
    if (invincible)
      {
       invincible_timer--;

            flash = !flash;  // Toggle flash every frame

            if (invincible_timer <= 0)
            {
           invincible = false;
           flash = false;
       }
    }

      // Apply movement
      x += hspeed;
      y += vspeed;

      // Handle facing direction and animation
      if (move != 0)
      {
            image_xscale = sign(move);
            sprite_index = sprite_run;
            image_speed = 1;
      }
      else
      {
            sprite_index = sprite_idle;
            image_speed = 0;
            image_index = 0;
      }

      // Ensure pixel-perfect positioning
      x = floor(x);
      y = floor(y);


      //Handle Room Moving
      if (x > room_width)
      {
            switch(room)
            {
                  // Tutorial Level Transitioning
                  case rm_tutorial_learn:
```

```
                        room_goto(rm_tutorial_encounter);
                        break;

                case rm_tutorial_encounter:
                        //global.current_section = 0;
                        //global.question_in_section = 0;
                        //global.number_of_questions = 0;
                        with(obj_heart_controller)
                        {
                                current_hearts = max_hearts;
                        }
                        room_goto(rm_level_complete);
                    break;

                     // First Level Transitioning
                     case rm_first_learn:
                            room_goto(rm_first_encounter);
                            break;

                case rm_first_encounter:
                    if ((global.question_in_section + 1) < global.number_of_questions)
                            {
                        global.question_in_section++;
                                show_debug_message("Incrementing question to " +
string(global.question_in_section + 1));
                        room_goto(rm_first_encounter);
                    }
                            else
                            {
                        // Section complete

                                if(global.current_section < 2) // If level is not
complete
                                {
                                        global.current_section++;  // Increment
Section
                                        global.question_in_section = 0;
                                        //global.number_of_questions = 0;
                                        show_debug_message("Incrementing section to "
+ string(global.current_section));
                                        room_goto(rm_first_learn); // Go to next
learn page
                                }
                                else // Go to next level
                                {
                                        global.current_section = 0;
                                        global.question_in_section = 0;
                                        global.number_of_questions = 0;
                                        with(obj_heart_controller)
                                        {
                                                current_hearts = max_hearts;
                                        }
                                        room_goto(rm_level_complete);
                                }
                    }
```

```
                    break;


                     //Second Level Transitioning
                     case rm_second_learn:
                           room_goto(rm_second_encounter);
                           break;

               case rm_second_encounter:
                  if ((global.question_in_section + 1) < global.number_of_questions)
                           {
                       global.question_in_section++;
                                 show_debug_message("Incrementing question to " +
string(global.question_in_section + 1));
                       room_goto(rm_second_encounter);
                    }
                           else
                           {
                       // Section complete

                                 if(global.current_section < 2) // If level is not
complete
                                 {
                                       global.current_section++;  // Increment
Section
                                       global.question_in_section = 0;
                                       //global.number_of_questions = 0;
                                       show_debug_message("Incrementing section to "
+ string(global.current_section));
                                       room_goto(rm_second_learn); // Go to next
learn page
                                 }
                                 else // Go to next level
                                 {
                                       global.current_section = 0;
                                       global.question_in_section = 0;
                                       global.number_of_questions = 0;
                                       with(obj_heart_controller)
                                       {
                                             current_hearts = max_hearts;
                                       }
                                       room_goto(rm_level_complete);
                                 }
                    }
                    break;


                     // Third Level Transitioning
                     case rm_third_learn:
                           room_goto(rm_third_encounter);
                           break;

               case rm_third_encounter:
                  if ((global.question_in_section + 1) < global.number_of_questions)
                           {
```

```
                        global.question_in_section++;
                                show_debug_message("Incrementing question to " +
string(global.question_in_section + 1));
                        room_goto(rm_third_encounter);
                    }
                            else
                            {
                        // Section complete

                                if(global.current_section < 2) // If level is not
complete
                                {
                                       global.current_section++;  // Increment
Section
                                       global.question_in_section = 0;
                                       //global.number_of_questions = 0;
                                       show_debug_message("Incrementing section to "
+ string(global.current_section));
                                       room_goto(rm_third_learn); // Go to next
learn page
                                }
                                else // Go to next level
                                {
                                       global.current_section = 0;
                                       global.question_in_section = 0;
                                       global.number_of_questions = 0;
                                       with(obj_heart_controller)
                                       {
                                              current_hearts = max_hearts;
                                       }
                                       room_goto(rm_level_complete);
                                }
                    }
                    break;


                     //Fourth Level Transitioning
                     case rm_fourth_learn:
                            room_goto(rm_fourth_encounter);
                            break;

                case rm_fourth_encounter:
                    if ((global.question_in_section + 1) < global.number_of_questions)
                            {
                        global.question_in_section++;
                                show_debug_message("Incrementing question to " +
string(global.question_in_section + 1));
                        room_goto(rm_fourth_encounter);
                    }
                            else
                            {
                        // Section complete

                                if(global.current_section < 2) // If level is not
complete
```

```
                                {
                                        global.current_section++;  // Increment
Section
                                        global.question_in_section = 0;
                                        //global.number_of_questions = 0;
                                        show_debug_message("Incrementing section to "
+ string(global.current_section));
                                        room_goto(rm_fourth_learn); // Go to next
learn page
                                }
                                else // Go to next level
                                {
                                        global.current_section = 0;
                                        global.question_in_section = 0;
                                        global.number_of_questions = 0;
                                        with(obj_heart_controller)
                                        {
                                                current_hearts = max_hearts;
                                        }
                                        room_goto(rm_level_complete);
                                }
                        }
                        break;
                }
        }

}
else
{
    // Stop all movement
    hspeed = 0;
    vspeed = 0;
}
```

Draw.gml

```
// Draw Event of obj_alan
if (invincible && flash)
{
    // Draw semi-transparent when invincible and flashing
    draw_sprite_ext(sprite_index, image_index, x, y,
        image_xscale, image_yscale, image_angle, c_white, 0.5);
}
else
{
    // Normal draw
    draw_self();
}
```

obj_first_enemy/obj_second_enemy/obj_third_enemy/obj_fourth_enemy

Create.gml

```
// Create Event of obj_enemy
// Create Event of obj_enemy
move_speed = (global.current_section + 1) * 4;
moving_right = true;  // true for moving right, false for moving left
sprite_half_width = sprite_width/2;
image_speed = 0;
```

Step.gml

```
// Step Event of obj_enemy
if (global.can_move)
{
    // Set movement direction
    hspeed = moving_right ? move_speed : -move_speed;

    // Boundary checking
    if (x + hspeed < sprite_half_width)
    {
        x = sprite_half_width;
        moving_right = true;
    }
    if (x + hspeed > room_width - sprite_half_width)
    {
        x = room_width - sprite_half_width;
        moving_right = false;
    }

    // Apply movement
    x += hspeed;

    // Check collision with Alan
    var collision = instance_place(x, y, obj_alan);
    if (collision != noone && !collision.invincible)
       {
        // If Alan is above the enemy (jumping on head)
        if (collision.y < y - sprite_height/2)
             {
            // Destroy enemy
            instance_destroy();
        }

    }
}

else
{
    image_speed = 0;
    // Stop all movement
    //move_speed = 0;
}
```

obj_dark_overlay

Create.gml

```
// Create Event of obj_dark_overlay
image_alpha = 0.8;  // Adjust transparency (0 = clear, 1 = solid)
depth = -9999;      // Make sure it appears on top of everything except the detail
panel
```

DrawGUI.gml

```
// Draw Event of obj_dark_overlay
draw_set_color(c_black);
draw_set_alpha(image_alpha);
draw_rectangle(0, 0, room_width, room_height, false);
draw_set_alpha(1);  // Reset alpha
```

obj_detail_panel

Create.gml

```
depth = -10000;

// Learn images
section_sprites[1, 0] = spr_level1_section0;     // Level 1, Section 0
section_sprites[1, 1] = spr_level1_section1;     // Level 1, Section 1
section_sprites[1, 2] = spr_level1_section2;     // Level 1, Section 2

section_sprites[2, 0] = spr_level2_section0;     // Level 2, Section 0
section_sprites[2, 1] = spr_level2_section1;     // Level 2, Section 1
section_sprites[2, 2] = spr_level2_section2;     // Level 2, Section 2

section_sprites[3, 0] = spr_level3_section0;     // Level 3, Section 0
section_sprites[3, 1] = spr_level3_section1;     // Level 3, Section 1
section_sprites[3, 2] = spr_level3_section2;     // Level 3, Section 2

section_sprites[4, 0] = spr_level4_section0;     // Level 4, Section 0
section_sprites[4, 1] = spr_level4_section1;     // Level 4, Section 1
section_sprites[4, 2] = spr_level4_section2;     // Level 4, Section 2
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_normal);
draw_set_halign(fa_left);
draw_set_valign(fa_top);
draw_set_color(c_white);

//draw_text_ext(60, 70, board_content_text, 20, 1130);
draw_sprite(section_sprites[global.current_level, global.current_section], 0, 55,
65);
```

obj_answer_button

Create.gml

```
hovering = false;
clicked = false;
image_alpha = 1;
fading = false;

sprite_still = spr_button;
sprite_hover = spr_inverted_spr;
```

Step.gml

```
if (fading)
{
    image_alpha -= 0.05;  // Adjust speed as needed
    if (image_alpha <= 0)
       {
        instance_destroy();
    }
}
```

LeftPressed.gml

```
// Left Pressed Event
image_alpha = 1;
clicked = true;
```

LeftReleased.gml

```
sprite_index = sprite_hover;

if (hovering && clicked)
{
	clicked = false;

	if (is_correct)
	{
		global.can_move = true;

		// Start fading all buttons and question panel
		with(obj_answer_button)
		{
			fading = true;
		}

		with(obj_question_panel)
		{
			fading = true;
		}

		show_message("Correct! Squish that bug!");

		with(obj_heart_controller)
```

```
		{
			if(current_hearts < 5)
			{
				current_hearts += 1;
			}

			flash_hearts = true;
			flash_timer = 0;
		}

	}
	else
	{
		global.can_move = false;
		show_message("Incorrect! Try again");

		with(obj_heart_controller)
		{
			current_hearts -= 1;

			flash_hearts = true;
			flash_timer = 0;

			if(current_hearts <= 0)
			{
				room_goto(rm_game_over);
				global.current_encounter = 1;
				current_hearts = max_hearts;  // Reset hearts
			}
		}

	}
}
else
{
	clicked = false;
}
```

MouseEnter.gml

```
sprite_index = sprite_hover;
hovering = true;
```

MouseLeave.gml

```
// Mouse Leave Event
sprite_index = sprite_still;
hovering = false;
clicked = false;
```

Draw.gml

```
draw_set_alpha(image_alpha);
draw_set_alpha(1);
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_button);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_color(c_white);


//draw_text(x, y, text);
draw_text_ext(x, y, text, 20, 280);
```

obj_encounter_controller

Create.gml

```
// Create Event of obj_button_controller
//globalvar can_move;
global.can_move = false;


// Create question panel first (using your existing question panel object)
question = instance_create_layer(640, 75, "UILayer", obj_question_panel);

if(current_level != 0)
{
	global.number_of_questions =
array_length(obj_question_panel.questions[global.current_level,
global.current_section]); //Set number of questions for each section
}
else
{
	global.number_of_questions = 1;
}
show_debug_message("Number of questions: " + string(global.number_of_questions));

// Create three buttons using your existing button object
button1 = instance_create_layer(185, 306, "UILayer", obj_answer_button);
button1.image_xscale = 1.5; //1.152381
button1.image_yscale = 1.5; //1.316832

button2 = instance_create_layer(640, 306, "UILayer", obj_answer_button);
button2.image_xscale = 1.5;
button2.image_yscale = 1.5;

button3 = instance_create_layer(1095, 306, "UILayer", obj_answer_button);
button3.image_xscale = 1.5;
button3.image_yscale = 1.5;


// Arrays to store all answers

//Tutorial Question
```

```
answers[0, 0] =
[
                    ["BEN", false], // First Answer Choice
                    ["ALAN", true], // Second Answer Choice
                    ["CAROL", false] // Third Answer Choice
];

//Level 1
// Level 1, Section 0: Introduction and Main
answers[1, 0] =
[
       //First Question
       [
           [  // Answers for First variant
                    ["1,000 bytes", true], // First Answer Choice
                    ["1,000,000 bytes", false], // Second Answer Choice
                    ["1,000,000,000 bytes", false] // Third Answer Choice
              ],
               [     // Answers for Second variant
                    ["1,000 bytes", false], // First Answer Choice
                    ["1,000,000 bytes", true], // Second Answer Choice
                    ["1,000,000,000 bytes", false] // Third Answer Choice
              ],
               [     // Answers for Third variant
                    ["1,000 bytes", false], // First Answer Choice
                    ["1,000,000 bytes", false], // Second Answer Choice
                    ["1,000,000,000 bytes", true] // Third Answer Choice
              ]
       ],
       //Second Question
       [
           [  // Answers for First variant
                    ["Hard Drive", true], // First Answer Choice
                    ["CPU", false], // Second Answer Choice
                    ["Keyboard", false] // Third Answer Choice
              ],
               [     // Answers for Second variant
                    ["Hard Drive", false], // First Answer Choice
                    ["CPU", true], // Second Answer Choice
                    ["Keyboard", false] // Third Answer Choice
              ],
               [     // Answers for Third variant
                    ["Hard Drive", false], // First Answer Choice
                    ["CPU", false], // Second Answer Choice
                    ["Keyboard", true] // Third Answer Choice
              ]
       ],
       //Third Question
       [
           [  // Answers for First variant
                    ["Coding", true], // First Answer Choice
                    ["Analyzing", false], // Second Answer Choice
                    ["Designing", false] // Third Answer Choice
              ],
               [     // Answers for Second variant
```

```
                    ["Software", false], // First Answer Choice
                    ["Hardware", true], // Second Answer Choice
                    ["Computer Parts", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["Beginner ", false], // First Answer Choice
                    ["Case-Insensitive", false], // Second Answer Choice
                    ["Case-Sensitive", true] // Third Answer Choice
                ]
        ],
        //Fourth Question
        [
            [  // Answers for First variant
                    ["8 bits", true], // First Answer Choice
                    ["2 bits", false], // Second Answer Choice
                    ["1 bit", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["First Include", false], // First Answer Choice
                    ["Main Function", true], // Second Answer Choice
                    ["First Comment", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["CPU", false], // First Answer Choice
                    ["Mouse", false], // Second Answer Choice
                    ["Monitor", true] // Third Answer Choice
                ]
        ]
];

// Level 1, Section 1:  Variables and Expressions
answers[1, 1] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["int x;", true], // First Answer Choice
                    ["num x;", false], // Second Answer Choice
                    ["var x;", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["int num;", false], // First Answer Choice
                    ["double num;", true], // Second Answer Choice
                    ["decimal num;", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["letter c;", false], // First Answer Choice
                    ["character c;", false], // Second Answer Choice
                    ["char c;", true] // Third Answer Choice
                ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                    ["stdio.h", true], // First Answer Choice
```

```
                    ["math.h", false], // Second Answer Choice
                    ["memory.h", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["print", false], // First Answer Choice
                    ["printf", true], // Second Answer Choice
                    ["cout", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["%d", false], // First Answer Choice
                    ["%i", false], // Second Answer Choice
                    ["%lf", true] // Third Answer Choice
                ]
        ],
        //Third Question
        [
            [  // Answers for First variant
                    ["variables", true], // First Answer Choice
                    ["numbers", false], // Second Answer Choice
                    ["words", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["Period (.)", false], // First Answer Choice
                    ["Semicolon (;)", true], // Second Answer Choice
                    ["Colon (:)", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["x = 100.1;", false], // First Answer Choice
                    ["x = \'100\';", false], // Second Answer Choice
                    ["x = 100;", true] // Third Answer Choice
                ]
        ]
];

// Level 1, Section 2:  Input and Operations
answers[1, 2] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["Groupings", true], // First Answer Choice
                    ["Addition and Subtraction", false], // Second Answer Choice
                    ["Powers and Functions", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["Groupings", false], // First Answer Choice
                    ["Addition and Subtraction", true], // Second Answer Choice
                    ["Powers and Functions", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["5", false], // First Answer Choice
                    ["0", false], // Second Answer Choice
                    ["1", true] // Third Answer Choice
                ]
        ],
```

```
        //Second Question
        [
            [  // Answers for First variant
                    ["scanf(\"%d\", &i);", true], // First Answer Choice
                    ["scanf(\"%lf\", &num);", false], // Second Answer Choice
                    ["scanf(\"%c\", &letter);", false] // Third Answer Choice
               ],
                [     // Answers for Second variant
                    ["int", false], // First Answer Choice
                    ["lf", true], // Second Answer Choice
                    ["d", false] // Third Answer Choice
               ],
                [     // Answers for Third variant
                    ["End of the Sentence", false], // First Answer Choice
                    ["First Semicolon", false], // Second Answer Choice
                    ["First White Space", true] // Third Answer Choice
               ]
        ],
        //Third Question
        [
            [  // Answers for First variant
                    ["Right", true], // First Answer Choice
                    ["Left", false], // Second Answer Choice
                    ["Middle", false] // Third Answer Choice
               ],
                [     // Answers for Second variant
                    ["MEGS", false], // First Answer Choice
                    ["GEMS", true], // Second Answer Choice
                    ["SDLC", false] // Third Answer Choice
               ],
                [     // Answers for Third variant
                    ["Division", false], // First Answer Choice
                    ["Multiplication", false], // Second Answer Choice
                    ["Modulus", true] // Third Answer Choice
               ]
        ]
];


//Level 2
// Level 2, Section 0: Conditionals
answers[2, 0] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["&&", true], // First Answer Choice
                    ["!", false], // Second Answer Choice
                    ["||", false] // Third Answer Choice
               ],
                [     // Answers for Second variant
                    ["&&", false], // First Answer Choice
                    ["!", true], // Second Answer Choice
                    ["||", false] // Third Answer Choice
               ],
```

```
                [     // Answers for Third variant
                      ["&&", false], // First Answer Choice
                      ["!", false], // Second Answer Choice
                      ["||", true] // Third Answer Choice
                ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                      ["1", true], // First Answer Choice
                      ["0", false], // Second Answer Choice
                      ["-1", false] // Third Answer Choice
                ],
                [     // Answers for Second variant
                      ["1", false], // First Answer Choice
                      ["0", true], // Second Answer Choice
                      ["-1", false] // Third Answer Choice
                ],
                [     // Answers for Third variant
                      ["if-else", false], // First Answer Choice
                      ["switch", false], // Second Answer Choice
                      ["printf", true] // Third Answer Choice
                ]
        ],
        //Third Question
        [
            [  // Answers for First variant
                      ["Multiple", true], // First Answer Choice
                      ["One", false], // Second Answer Choice
                      ["None", false] // Third Answer Choice
                ],
                [     // Answers for Second variant
                      ["double", false], // First Answer Choice
                      ["character", true], // Second Answer Choice
                      ["float", false] // Third Answer Choice
                ],
                [     // Answers for Third variant
                      ["// (Double Slashes)", false], // First Answer Choice
                      ["() (Parenthesis)", false], // Second Answer Choice
                      ["{} (Curly Braces)", true] // Third Answer Choice
                ]
        ]
];

// Level 2, Section 1: Functions
answers[2, 1] =
[
        //First Question
        [
            [  // Answers for First variant
                      ["One", true], // First Answer Choice
                      ["Many", false], // Second Answer Choice
                      ["None", false] // Third Answer Choice
                ],
                [     // Answers for Second variant
```

```
                    ["One", false], // First Answer Choice
                    ["Many", true], // Second Answer Choice
                    ["None", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["Conditional", false], // First Answer Choice
                    ["Variable", false], // Second Answer Choice
                    ["Function", true] // Third Answer Choice
                ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                    ["Above Main", true], // First Answer Choice
                    ["Beneath Main", false], // Second Answer Choice
                    ["Anywhere in Main", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["Four", false], // First Answer Choice
                    ["Three", true], // Second Answer Choice
                    ["One", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["Square", false], // First Answer Choice
                    ["Variable", false], // Second Answer Choice
                    ["Double", true] // Third Answer Choice
                ]
        ]
];

// Level 2, Section 2: Loops
answers[2, 2] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["when some code will be run multiple times", true], // First
Answer Choice
                    ["in the beginning of programs", false], // Second Answer Choice
                    ["at the end of programs", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["do-while", false], // First Answer Choice
                    ["do-until", true], // Second Answer Choice
                    ["while", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["Even", false], // First Answer Choice
                    ["False", false], // Second Answer Choice
                    ["True", true] // Third Answer Choice
                ]
        ],
        //Second Question
        [
            [  // Answers for First variant
```

```
                    ["do-while", true], // First Answer Choice
                    ["while", false], // Second Answer Choice
                    ["for", false] // Third Answer Choice
             ],
              [     // Answers for Second variant
                    ["do-while", false], // First Answer Choice
                    ["while", true], // Second Answer Choice
                    ["for", false] // Third Answer Choice
             ],
              [     // Answers for Third variant
                    ["do-while", false], // First Answer Choice
                    ["while", false], // Second Answer Choice
                    ["for", true] // Third Answer Choice
             ]
      ],
      //Third Question
      [
          [  // Answers for First variant
                    ["for (n=1; n<11; n++)\n   printf (\"hi\");", true], // First
Answer Choice
                    ["while (n < 10)\n    n = n + 1;", false], // Second Answer
Choice
                    ["for (n=0; n<=10; n++)\n   printf (\"bye\");", false] // Third
Answer Choice
             ],
              [     // Answers for Second variant
                    ["always needed", false], // First Answer Choice
                    ["to execute multiple commands", true], // Second Answer Choice
                    ["to run just once", false] // Third Answer Choice
             ],
              [     // Answers for Third variant
                    ["verify; check; modify", false], // First Answer Choice
                    ["start; condition; end", false], // Second Answer Choice
                    ["initialize; condition; increment", true] // Third Answer Choice
             ]
      ]
];


//Level 3
// Level 3, Section 0: Arrays
answers[3, 0] =
[
      //First Question
      [
          [  // Answers for First variant
                    ["4", true], // First Answer Choice
                    ["37", false], // Second Answer Choice
                    ["2000", false] // Third Answer Choice
             ],
              [     // Answers for Second variant
                    ["1", false], // First Answer Choice
                    ["0", true], // Second Answer Choice
                    ["-1", false] // Third Answer Choice
             ],
```

```
                [     // Answers for Third variant
                      ["4", false], // First Answer Choice
                      ["37", false], // Second Answer Choice
                      ["2000", true] // Third Answer Choice
                ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                      ["for", true], // First Answer Choice
                      ["while", false], // Second Answer Choice
                      ["do-while", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                      ["{} (Curly Braces)", false], // First Answer Choice
                      ["[] (Brackets)", true], // Second Answer Choice
                      ["() (Parenthesis)", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                      ["val", false], // First Answer Choice
                      ["int", false], // Second Answer Choice
                      ["3", true] // Third Answer Choice
                ]
        ],
        //Third Question
        [
            [  // Answers for First variant
                      ["char", true], // First Answer Choice
                      ["array", false], // Second Answer Choice
                      ["int", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                      ["arrayName[50][double];", false], // First Answer Choice
                      ["double arrayName[50];", true], // Second Answer Choice
                      ["arrayName[50] = double;", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                      ["number int[7];", false], // First Answer Choice
                      ["int[7] numbers;", false], // Second Answer Choice
                      ["int numbers[7];", true] // Third Answer Choice
                ]
        ]
];

// Level 3, Section 1: Strings
answers[3, 1] =
[
        //First Question
        [
            [  // Answers for First variant
                      ["\\0", true], // First Answer Choice
                      ["\\n", false], // Second Answer Choice
                      ["\"", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
```

```
            ["char", false], // First Answer Choice
            ["string", true], // Second Answer Choice
            ["function", false] // Third Answer Choice
        ],
         [     // Answers for Third variant
            ["123", false], // First Answer Choice
            ["'123'", false], // Second Answer Choice
            ["\"123\"", true] // Third Answer Choice
        ]
],
//Second Question
[
    [  // Answers for First variant
            ["\"\" (Double Quotes)", true], // First Answer Choice
            ["() (Parenthesis)", false], // Second Answer Choice
            ["'' (Single Quotes)", false] // Third Answer Choice
        ],
         [     // Answers for Second variant
            ["stdio.h", false], // First Answer Choice
            ["string.h", true], // Second Answer Choice
            ["stdin.h", false] // Third Answer Choice
        ],
         [     // Answers for Third variant
            ["strlen", false], // First Answer Choice
            ["strcat", false], // Second Answer Choice
            ["strcmp", true] // Third Answer Choice
        ]
],
//Third Question
[
    [  // Answers for First variant
            ["strlen", true], // First Answer Choice
            ["strcat", false], // Second Answer Choice
            ["strcmp", false] // Third Answer Choice
        ],
         [     // Answers for Second variant
            ["strlen", false], // First Answer Choice
            ["strcat", true], // Second Answer Choice
            ["strcmp", false] // Third Answer Choice
        ],
         [     // Answers for Third variant
            ["strncpy", false], // First Answer Choice
            ["strcmp", false], // Second Answer Choice
            ["strcpy", true] // Third Answer Choice
        ]
],
//Fourth Question
[
    [  // Answers for First variant
            ["strncpy", true], // First Answer Choice
            ["strcmp", false], // Second Answer Choice
            ["strcpy", false] // Third Answer Choice
        ],
         [     // Answers for Second variant
            ["12", false], // First Answer Choice
```

```
                    ["11", true], // Second Answer Choice
                    ["10", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["1", false], // First Answer Choice
                    ["-1", false], // Second Answer Choice
                    ["0", true] // Third Answer Choice
                ]
        ]
];

// Level 3, Section 2: Addresses/Pointers
answers[3, 2] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["* (Asterisk)", true], // First Answer Choice
                    ["& (Ampersand)", false], // Second Answer Choice
                    ["% (Modulus)", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["* (Asterisk)", false], // First Answer Choice
                    ["& (Ampersand)", true], // Second Answer Choice
                    ["% (Modulus)", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["File Path", false], // First Answer Choice
                    ["Location on Desktop", false], // Second Answer Choice
                    ["Location in RAM", true] // Third Answer Choice
                ]
        ]
];


//Level 4
// Level 4, Section 0: Enumerate and Structures
answers[4, 0] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["enum", true], // First Answer Choice
                    ["struct", false], // Second Answer Choice
                    ["arr", false] // Third Answer Choice
                ],
                 [     // Answers for Second variant
                    ["* (Asterisk)", false], // First Answer Choice
                    [". (Period)", true], // Second Answer Choice
                    ["& (Ampersand)", false] // Third Answer Choice
                ],
                 [     // Answers for Third variant
                    ["array (arr)", false], // First Answer Choice
                    ["integer (int)", false], // Second Answer Choice
                    ["structure (struct)", true] // Third Answer Choice
```

```
            ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                    ["True", true], // First Answer Choice
                    ["False", false], // Second Answer Choice
                    ["Maybe", false] // Third Answer Choice
            ],
             [     // Answers for Second variant
                    ["True", false], // First Answer Choice
                    ["False", true], // Second Answer Choice
                    ["Maybe", false] // Third Answer Choice
            ],
             [     // Answers for Third variant
                    ["True", true], // First Answer Choice
                    ["False", false], // Second Answer Choice
                    ["Maybe", false] // Third Answer Choice
            ]
        ]
];

// Level 4, Section 1: File Input and Output
answers[4, 1] =
[
        //First Question
        [
            [  // Answers for First variant
                    ["fopen", true], // First Answer Choice
                    ["fclose", false], // Second Answer Choice
                    ["fgets", false] // Third Answer Choice
            ],
             [     // Answers for Second variant
                    ["fopen", false], // First Answer Choice
                    ["fclose", true], // Second Answer Choice
                    ["fgets", false] // Third Answer Choice
            ],
             [     // Answers for Third variant
                    ["printf", false], // First Answer Choice
                    ["fprint", false], // Second Answer Choice
                    ["fprintf", true] // Third Answer Choice
            ]
        ],
        //Second Question
        [
            [  // Answers for First variant
                    ["fgets", true], // First Answer Choice
                    ["fclose", false], // Second Answer Choice
                    ["fopen", false] // Third Answer Choice
            ],
             [     // Answers for Second variant
                    ["*afile FILE;", false], // First Answer Choice
                    ["FILE *afile;", true], // Second Answer Choice
                    ["afile *file;", false] // Third Answer Choice
            ],
```

```
              [     // Answers for Third variant
                    ["-", false], // First Answer Choice
                    ["/", false], // Second Answer Choice
                    ["\\\\", true] // Third Answer Choice
              ]
       ]
];

// Level 4, Section 2: Object Oriented Programming
answers[4, 2] =
[
       //First Question
       [
           [  // Answers for First variant
                    ["Inheritence; Abstraction; Encapsulation; Polymorphism", true],
// First Answer Choice
                    ["Analyze; Design; Code; Test", false], // Second Answer Choice
                    ["Input; Output; Process; Storage", false] // Third Answer Choice
             ],
              [     // Answers for Second variant
                    ["A variable that stores a number.", false], // First Answer
Choice
                    ["Anything that has attributes and can perform methods.", true],
// Second Answer Choice
                    ["A string that states the name and location of a variable.",
false] // Third Answer Choice
             ],
              [     // Answers for Third variant
                    ["Efficiency", false], // First Answer Choice
                    ["Reusability", false], // Second Answer Choice
                    ["Simpler", true] // Third Answer Choice
              ]
       ]
];

// Set Up Answers
if(current_level != 0)
{
       var current_answers = answers[global.current_level,
global.current_section][global.question_in_section][question.variation - 1];
}
else
{
       var current_answers = answers[0, 0];
}

button1.text = current_answers[0][0];
button1.is_correct = current_answers[0][1];

button2.text = current_answers[1][0];
button2.is_correct = current_answers[1][1];

button3.text = current_answers[2][0];
button3.is_correct = current_answers[2][1];
```

obj_question_panel

Create.gml

```
// Add alpha variable for fading
image_alpha = 1;
fading = false;


randomize();
variation = irandom_range(1, 3);  // Random number between 1 and 3
//variation = 3; //Testing each variation

// Question data structure

// Tutorial Question
questions[0, 0] = "What is your name?\n (Hint: It is the name of the game!)";

//Level 1
// Level 1, Section 0: Introduction and Main
questions[1, 0] =
[
	[	// First Question
		"How many bytes are in a KB?", // First variant
		"How many bytes are in a MB?", // Second variant
		"How many bytes are in a GB?" // Third variant
	],
	[	// Second Question
		"What part of a computer manipulates and stores data?", // First variant
		"What part of a computer processes data?", // Second variant
		"What part of a computer receives input?" // Third variant
	],
	[	// Third Question
		"The third phase of software engineering that we are focusing on is
_______.", // First variant
		"The physical parts of the computer are called _______.", // Second
variant
		"What type of language is C?" // Third variant
	],
	[	// Fourth Question
		"How many bits are in a byte?", // First variant
		"When running, every C program has and starts executing at the
_______.", // Second variant
		"What part of a computer presents output?" // Third variant
	]
];

// Level 1, Section 1: Variables and Expressions
questions[1, 1] =
[
	[	// First Question
		"In C, the syntax to declare an integer variable called x is", // First
variant
		"In C, the syntax to declare a decimal variable called num is", //
Second variant
```

```
            "In C, the syntax to declare a character variable called c is" // Third
variant
      ],
      [     // Second Question
            "Which header file should be included to get access to input and output
functions?", // First variant
            "Which is the C command to show text on the screen/monitor?", // Second
variant
            "What place holder in a format string will print out a double/decimal
value?" // Third variant
      ],
      [     // Third Question
            "In order to read in and manipulate data, we need _______ to store
values.", // First variant
            "Each statement in a C program should end with what?", // Second variant
            "Which statement is a complete assignment of an integer value to a
variable called x?" // Third variant
      ]
];

// Level 1, Section 2: Input and Operations
questions[1, 2] =
[
      [     // First Question
            "In programming, what is the highest order of operations?", // First
variant
            "In programming, what is the lowest order of operations?", // Second
variant
            "What does this expression evaluate to? \n 4 * 3 / 6 - 5 % 3 + 1" //
Third variant
      ],
      [     // Second Question
            "Which line reads an integer into a variable?", // First variant
            "Fill in the blank to read a decimal number into a variable:
scanf(\"&__\",&number);", // Second variant
            "The scanf command reads up to the _______." // Third variant
      ],
      [     // Third Question
            "In programming, what side of an assignment is evaluated first?", //
First variant
            "What is one acronym for programming order of operations?", // Second
variant
            "In a C expression, what is %?" // Third variant
      ]
];


//Level 2
// Level 2, Section 0: Conditionals
questions[2, 0] =
[
      [     // First Question
            "Which is the C symbol for logical 'and'?", // First variant
            "Which is the C symbol for logical 'not'?", // Second variant
            "Which is the C symbol for logical 'or'?" // Third variant
```

```
	],
	[	// Second Question
		"Which numerical value is considered 'true'?", // First variant
		"Which numerical value is considered 'false'?", // Second variant
		"Which is NOT a conditional statement?" // Third variant
	],
	[	// Third Question
		"A switch function can have _______ case values.", // First variant
		"The expression at the start of a switch statement must evaluate to an
integer or a _______.", // Second variant
		"What are the multiple statements of code to be run in a conditional
contained in?" // Third variant
	]
];

// Level 2, Section 1: Functions
questions[2, 1] =
[
	[	// First Question
		"How many values can be returned from a function?", // First variant
		"How many variables can a function take in?", // Second variant
		"A _______ is a set of lines of code that can be called to be executed
from another location in a program and may have inputs and an output." // Third
variant
	],
	[	// Second Question
		"Typically, in C, functions that you will write should be _______ unless
you use prototypes.", // First variant
		"A function that starts out with \"int addition (int a, int b, int c)\"
will take in how many values?", // Second variant
		"A function that starts with \"double squareRoot (double x, double y)\"
is called a what type function?" // Third variant
	]
];

// Level 2, Section 2: Loops
questions[2, 2] =
[
	[	// First Question
		"When are loops used?", // First variant
		"Which is NOT a loop in the C language?", // Second variant
		"In a loop, the code will execute as long as the condition is _______."
// Third variant
	],
	[	// Second Question
		"If code in a loop will be executed at least one time, then a _______
loop may be more appropriate.", // First variant
		"Which is the fundamental loop for most programming languages?", //
Second variant
		"Which type of loop is often used to walk through each element of an
array?" // Third variant
	],
	[	// Third Question
		"Assuming \"int n=1;\" is already declared, which loop will run exactly
10 times?", // First variant
```

```
            "When do loop (and if-else) statements need '{' and '}'?", // Second
variant
            "What are the three things in the start of a for-loop?\n for (____;
____; ____)" // Third variant
      ]
];


//Level 3
// Level 3, Section 0: Arrays
questions[3, 0] =
[
      [     // First Question
            "How many dimensions is this array?\n char multiDim[5][10][2][20];", //
First variant
            "What is the index of the first position in an array?", // Second
variant
            "How many characters can be stored in this array?\n char
multiDim[5][10][2][20];" // Third variant
      ],
      [     // Second Question
            "Which type of loop is usually used to walk through each element of an
array?", // First variant
            "What symbols go in this array declaration?\n double anArray__30__;", //
Second variant
            "Fill in the blank in the following line of code creating an array:\n
int anArray [____] = {val1, val2, val3};" // Third variant
      ],
      [     // Third Question
            "Fill in the blank in the following line of code creating an array:\n
____ arrayName [2] = {'F', 'T'};", // First variant
            "Which is the correct way to create an array of floating-point values of
length 50?", // Second variant
            "What is the correct way to declare an integer array with 7 values?" //
Third variant
      ]
];

// Level 3, Section 1: Strings
questions[3, 1] =
[
      [     // First Question
            "Which symbol is used for the end of a string for printing, also called
the null character?", // First variant
            "A character array in C can be treated as a _______ or set of words.",
// Second variant
            "Which of the following is a string value in C?" // Third variant
      ],
      [     // Second Question
            "A constant string value has _______ around it.", // First variant
            "What is the name of header file to give access to string functions?",
// Second variant
            "Which command is used to compare strings?" // Third variant
      ],
      [     // Third Question
```

```
            "Which command returns the length of a string?", // First variant
            "Which command is used to append two strings together?", // Second
variant
            "Which command copies the value of one string to another string?" //
Third variant
      ],
      [     // Fourth Question
            "Which command copies a substring into another string?", // First
variant
            "What value is returned by strlen for: \n char astring[30] =
\"programming\";\n strlen (astring);", // Second variant
            "What value does strcmp return if the two string values are the same?"
// Third variant
      ]
];

// Level 3, Section 2: Addresses/Pointers
questions[3, 2] =
[
      [     // First Question
            "Fill in the blank for this pointer declaration:\n int ___ptrNum;", //
First variant
            "Which symbol is used in front of a variable name to mean the address of
the variable?", // Second variant
            "The left-hand side of any assignment statement in C must be a _______."
// Third variant
      ]
];


//Level 4
// Level 4, Section 0: Enumerate and Structures
questions[4, 0] =
[
      [     // First Question
            "Which declaration command can be used to create words as values?", //
First variant
            "What operator is used to access fields in a structure?", // Second
variant
            "What variable type allows multiple type values in it?" // Third variant
      ],
      [     // Second Question
            "In an enum, the values behind the words are integers.", // First
variant
            "Enum values cannot be used in case values in switch statements.", //
Second variant
            "You can create arrays of structures." // Third variant
      ]
];

// Level 4, Section 1: File Input and Output
questions[4, 1] =
[
      [     // First Question
            "Which command is used to open a file?", // First variant
```

```
            "For every fopen command in a program, there should be an _______
command?", // Second variant
            "Which command writes text to a file?" // Third variant
      ],
      [     // Second Question
            "Which command reads lines of text from a file?", // First variant
            "Select the code that declares a pointer to a file?", // Second variant
            "To find a location of file with its path, in every place there is a \,
we need to change it to ____." // Third variant
      ]
];

// Level 4, Section 2: Object Oriented Programming
questions[4, 2] =
[
      [     // First Question
            "What are the 4 methods in OOP?", // First variant
            "What is an Object in OOP?", // Second variant
            "What is not an advantage of OOP?" // Third variant
      ]
];


// Set up questions
if(current_level != 0)
{
      question_text = questions[global.current_level,
global.current_section][global.question_in_section][variation - 1];
}
else
{
      question_text = questions[0, 0];
}
```

Step.gml

```
if (fading)
{
    image_alpha -= 0.05;  // Adjust speed as needed
    if (image_alpha <= 0)
       {
        instance_destroy();
    }
}
```

Draw.gml

```
// Draw Event
draw_set_alpha(image_alpha);
draw_set_alpha(1);
```

DrawGUI.gml

```
draw_self();
```

```
draw_set_font(fnt_question);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_color(c_white);

//draw_text(x, y, board_question);
draw_text_ext(room_width/2, y, question_text, 40, 700);  // Line height 20, max width
360
```

obj_dying_alan

Create.gml

```
sprite_index = spr_alan_idle;
image_speed = 1;  // Control animation speed
image_xscale = 1.5;  // Scale if needed
image_yscale = 1.5;
animation_stage = 0;
timer = 0;

// Draw Event
draw_self();
```

Step.gml

```
timer++;

switch(animation_stage)
{
    case 0:  // Initial fall
        if (timer > 120)
             {
            animation_stage = 1;
            timer = 0;
        }
        y += 4;  // Slowly fall
        break;

    case 1:  // Flash
             image_xscale = 1.5;  // Scale if needed
             image_yscale = 1.5;
        if (timer mod 20 < 10)
             {  // Flash every few frames
            sprite_index = spr_alan_idle;
        }
             else
             {
            sprite_index = spr_reverse_idle
        }

        if (timer > 120)
             {  // After 2 second of flashing
                    sprite_index = spr_reverse_idle
            animation_stage = 2;
```

```
            timer = 0;
        }
        break;

    case 2:  // Final Stay
        animation_stage = 3;
        break;
}
```

obj_game_over_controller

Create.gml

```
// obj_game_over_controller Create Event
alan_death = instance_create_layer(room_width/2, 0, "UILayer", obj_dying_alan);


text_alpha = 0;  // Start fully transparent
button_alpha = 0;

// Wait until animation is done to show button
show_buttons = false;
alarm[0] = 240;  // 6 seconds

// Create back button
goBackButton = instance_create_layer(room_width/2, 625, "UILayer", obj_start_button);
goBackButton.image_xscale = 1.152381;
goBackButton.image_yscale = 1.316832;
goBackButton.text = "Go Back";
goBackButton.button_type = "select_level";
goBackButton.image_alpha = 0;
goBackButton.visible = false;
// Draw Event
draw_self();
```

Step.gml

```
if (alan_death.animation_stage >= 2)
{
    if (text_alpha < 1)
       {
        text_alpha += 0.01;  // Adjust speed as needed
    }
}

// Start fading buttons after show_buttons is true
if (show_buttons)
{
    if (button_alpha < 1)
       {
        button_alpha += 0.001;  // Adjust speed as needed
             goBackButton.image_alpha += button_alpha;
             if (!goBackButton.visible) goBackButton.visible = true;
    }
```

```
}
```

Alarm0.gml

```
// Alarm 0 Event
show_buttons = true;
```

DrawGUI.gml

```
draw_set_font(fnt_title);
draw_set_color(c_red);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);

// Draw Game Over
draw_set_alpha(text_alpha);
draw_text(room_width/2, 100, "Game Over");

draw_set_color(c_white);

// Reset alpha for other drawing
draw_set_alpha(1);
```

obj_ground_tutorial /obj_ground_first/obj_ground_second/ obj_ground_third/ obj_ground_fourth

Create.gml

```
// Ensure pixel-perfect positioning
y = floor(y);
```

Draw.gml

```
// Add this to obj_ground's Draw Event temporarily for debugging
draw_self();
draw_rectangle(bbox_left, bbox_top, bbox_right, bbox_bottom, true);
```

obj_learn_controller

Create.gml

```
global.can_move = true;
panel = instance_create_layer(640, 75, "UILayer", obj_learn_panel);
```

obj_learn_panel

Create.gml

```
hovering = false;
clicked = false;
expanded = false;

// Question data structure
//Tutorial Level
```

```
board[0, 0] = "Click Here for Tutorial Tips 1"; // First section
board[0, 1] = "Click Here for Tutorial Tips 2"; // Second section
board[0, 2] = "Click Here for Tutorial Tips 3"; // Third section

//First Level
board[1, 0] = "Click Here for C Tips: Introduction and Main"; // First section
board[1, 1] = "Click Here for C Tips: Variables and Expressions"; // Second section
board[1, 2] = "Click Here for C Tips: Input and Operations"; // Third section

//Second Level
board[2, 0] = "Click Here for C Tips: Conditionals"; // First section
board[2, 1] = "Click Here for C Tips: Functions"; // Second section
board[2, 2] = "Click Here for C Tips: Loops"; // Third section

//Third Level
board[3, 0] = "Click Here for C Tips: Arrays"; // First section
board[3, 1] = "Click Here for C Tips: Strings - Character Arrays"; // Second section
board[3, 2] = "Click Here for C Tips: Addresses and Pointers"; // Third section

//Fourth Level
board[4, 0] = "Click Here for C Tips: Enumerate and Structures"; // First section
board[4, 1] = "Click Here for C Tips: File Input and Output"; // Second section
board[4, 2] = "Click Here for C Tips: Object-Oriented Programming"; // Third section

board_text = board[global.current_level, global.current_section];
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_question);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);


if (hovering == true)
{
	draw_set_color(c_yellow);
	draw_text(room_width/2, y, board_text);
}
else
{
	draw_set_color(c_white);
}

draw_text(room_width/2, y, board_text);
```

LeftPressed.gml

```
clicked = true;
```

LeftReleased.gml

```
if (hovering && clicked)
{
```

```
	if (!expanded)
	{
		clicked = false;
		// Create dark overlay first (so it's behind the detail panel)
		instance_create_layer(0, 0, "UILayer", obj_dark_overlay);

		extendedView = instance_create_layer(room_width/2, room_height/2,
"UILayer", obj_detail_panel);
		extendedView.image_yscale = 4.5;

		backButton = instance_create_layer(1190, 80, "UILayer",
obj_back_button);
		backButton.backButton_type = "close_detail";

		expanded = true;

		visible = false;
		global.can_move = false;

	}
	else
	{
		expanded = false;
		visible = true;
		global.can_move = true;
	}

}
else
{
	clicked = false;

}
```

MouseEnter.gml

```
hovering = true;
```

MouseLeave.gml

```
hovering = false;
clicked = false;


draw_text(room_width/2, y, board_text);
draw_set_color(c_white)
```

obj_level_button

Create.gml

```
// obj_level_button Create Event
level_type = -1;  // Will be set according to global.current_level values
show_debug_message("Button level: " + string(level_type));
show_debug_message("Highest level: " + string(global.highest_level));
```

```
unlocked = (level_type <= global.highest_level);
show_debug_message("Button unlocked: " + string(unlocked));
selected = false;

hovering = false;
clicked = false;
```

Draw.gml

```
// Draw Event
draw_self();

// Draw the border circle first
if (!unlocked)
{
    draw_set_color(c_gray);
}
else if (selected)
{
    draw_set_color(c_red);
}
else
{
    draw_set_color(c_white);
}

// Draw filled circle slightly larger than the sprite
draw_circle(x - 1, y, sprite_width/2 + 7, false);

if (!unlocked)
{
      draw_set_alpha(0.7);
      draw_set_color(c_gray);
    draw_sprite_ext(sprite_index, 0, x, y, image_xscale, image_yscale, 0, c_gray, 1);

      draw_set_alpha(1);
    draw_text(x, y + 100 , "Locked");

      // Reset alpha and color after drawing this button
      draw_set_color(c_white);

}
else
{
      draw_set_color(c_white);
      draw_set_font(fnt_select);
      draw_sprite_ext(sprite_index, 0, x, y, image_xscale, image_yscale, 0, c_white,
1);


      switch(level_type)
      {
            case 0:
                  draw_text(x, y + 100, "Tutorial");
                  break;
```

```
		case 1:
			draw_text(x, y + 100, "Fundamentals");
			break;
		case 2:
			draw_text(x, y + 100, "Control Flows");
			break;
		case 3:
			draw_text(x, y + 100, "Data Structures");
			break;
		case 4:
			draw_text(x, y + 100, "Complex Concepts");
			break;
	}

	draw_set_color(c_white);
}
```

LeftPressed.gml

```
clicked = true;
```

LeftReleased.gml

```
if (hovering && clicked)
{
	clicked = false;

	if (unlocked)
	{
		// Deselect all other buttons first
		with (obj_level_button)
		{
			selected = false;
		}

		// Select this button
		selected = true;
	}

}
else
{
	clicked = false;
}
```

MouseEnter.gml

```
hovering = true;
```

MouseLeave.gml

```
hovering = false;
clicked = false;
```

obj_complete_controller

Create.gml

```
/// obj_level_complete_controller Create Event

// Save progress
show_debug_message("Before save - Highest level: " + string(global.highest_level));
show_debug_message("Current level: " + string(global.current_level));

global.highest_level = max(global.highest_level, global.current_level + 1);
show_debug_message("After update - Highest level: " + string(global.highest_level));


with(obj_game_controller)
{
	save_game();
}

// Set text based on which level was completed
switch(global.current_level)
{
	case 0: // Tutorial
	  completed_text = "You've passed Tutorial";
	  next_text = " is now unlocked!";
	  next_logo = spr_fundamental_btn;
	  break;
    case 1: // First
        completed_text = "You've mastered Fundamentals";
        next_text = " is now unlocked!";
        next_logo = spr_control_btn;
        break;
    case 2: // Second
        completed_text = "You've mastered Control Flows";
        next_text = " is now unlocked!";
        next_logo = spr_data_btn;
        break;
    case 3: // Third
        completed_text = "You've mastered Data Structures";
        next_text = " is now unlocked!";
        next_logo = spr_complex_btn;
        break;
    case 4: // Fourth
        completed_text = "You've mastered Java!";
        next_text = "Congratulations on completing all levels!";
        next_logo = noone;
             global.game_complete = true;
             with(obj_game_controller)
             {
                   save_game();
             }
        break;
}

// Create back button
goBackButton = instance_create_layer(room_width/2, 625, "UILayer", obj_start_button);
goBackButton.image_xscale = 1.152381;
```

```
goBackButton.image_yscale = 1.316832;
goBackButton.text = "Go Back";
goBackButton.button_type = "select_level";
```

DrawGUI.gml

```
// Draw GUI Event
// Draw GUI Event
draw_set_font(fnt_title);
draw_set_color(c_white);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);

// Draw congratulations
draw_text(room_width/2, 100, "Congratulations!");


// Draw completion text
draw_set_font(fnt_smallTitle);
draw_text(room_width/2, 250, completed_text);

// Draw next level logo if there is one
if (next_logo != noone)
{
    draw_sprite(next_logo, 0, room_width/2 - 250, room_height/2 + 48);
}
// Draw next level text in red
draw_set_color(c_red);
draw_text(room_width/2 + 50, room_height/2 + 50, next_text);

draw_set_font(fnt_title);
draw_set_color(c_white);
```

obj_level_controller

Create.gml

```
// obj_level_select_controller Create Event

// Create back button
backButton = instance_create_layer(64, 64, "UILayer", obj_back_button);
backButton.backButton_type = "start";

// Create back button
trophyButton = instance_create_layer(1120, 32, "UILayer", obj_trophy_button);


//Create Tutorial button
tutorialButton = instance_create_layer(260, 310, "UILayer", obj_level_button);
tutorialButton.level_type = 0;
tutorialButton.sprite_index = spr_tutorial_btn;
tutorialButton.unlocked = true;
tutorialButton.selected = true;
```

```
//Create First Level button
firstLevelButton = instance_create_layer(450, 310, "UILayer", obj_level_button);
firstLevelButton.level_type = 1;
firstLevelButton.sprite_index = spr_fundamental_btn;
firstLevelButton.unlocked = (global.highest_level >= 1);
//firstLevelButton.selected = true;


//Create Second Level button
secondLevelButton = instance_create_layer(640, 310, "UILayer", obj_level_button);
secondLevelButton.level_type = 2;
secondLevelButton.sprite_index = spr_control_btn;
secondLevelButton.unlocked = (global.highest_level >= 2);


//Create Third Level button
thirdLevelButton = instance_create_layer(830, 310, "UILayer", obj_level_button);
thirdLevelButton.level_type = 3;
thirdLevelButton.sprite_index = spr_data_btn;
thirdLevelButton.unlocked = (global.highest_level >= 3);


//Create Fourth Level button
fourthLevelButton = instance_create_layer(1020, 310, "UILayer", obj_level_button);
fourthLevelButton.level_type = 4;
fourthLevelButton.sprite_index = spr_complex_btn;
fourthLevelButton.unlocked = (global.highest_level >= 4);


// Create play button
playButton = instance_create_layer(room_width/2, room_height - 100, "UILayer",
obj_level_play_button);
playButton.image_xscale = 1.152381;
playButton.image_yscale = 1.316832;
playButton.text = "Play";
```

DrawGUI.gml

```
// Title
draw_set_font(fnt_title);
draw_set_halign(fa_center);
draw_text(room_width/2, 100, "Select Level");

//Reset
draw_set_font(fnt_button);
```

obj_level_play_button

Create.gml

```
// obj_play_button Create Event
hovering = false;
clicked = false;
```

```
sprite_still = spr_button;
sprite_hover = spr_inverted_spr;
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_button);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_color(c_white);

//draw_text(x, y, text);
draw_text_ext(x, y, text, 20, 175);
```

LeftPressed.gml

```
sprite_index = sprite_still;
clicked = true;
```

LeftReleased.gml

```
sprite_index = sprite_hover;

if (hovering && clicked)
{
	clicked = false;

	var selected_level = 0;

	// Find selected level
	with(obj_level_button)
	{
		if (selected && unlocked)
		{
			selected_level = level_type;
			break;
		}
	}

	global.current_level = selected_level;

	switch(selected_level)
	{
	  case 0: // Tutorial
	      room_goto(rm_tutorial_learn);
			show_debug_message("Tutorial");
	      break;
	  case 1: // First
	      room_goto(rm_first_learn);
			show_debug_message(selected_level);
	      break;
	  case 2: // Second
	      room_goto(rm_second_learn);
			show_debug_message(selected_level);
```

```
            break;
             case 3: // Third
            room_goto(rm_third_learn);
                    show_debug_message(selected_level);
            break;
             case 4: // Fourth
            room_goto(rm_fourth_learn);
                    show_debug_message(selected_level);
            break;
    }
}
else
{
       clicked = false;
}
```

MouseEnter.gml

```
sprite_index = sprite_hover;
hovering = true;
```

MouseLeave.gml

```
sprite_index = sprite_still;
hovering = false;
clicked = false;
```

obj_start_alan

Create.gml

```
// obj_start_alan Create Event
sprite_index = spr_alan_moving;
image_speed = 1;  // Control animation speed
image_xscale = 2;  // Scale if needed
image_yscale = 2;

// No need for Step Event as we just want the running animation

// Draw Event
draw_self();
```

DrawGUI.gml

```
// Create Event
title_y = 100;  // Adjust this value for height position

// Draw GUI Event
draw_set_font(fnt_title);  // Create and use a large font for the title
draw_set_color(c_white);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_text(room_width/2, title_y, "Alan's Coding Journey");
draw_set_halign(fa_left);  // Reset alignment
draw_set_valign(fa_top);
```

obj_start_button

Create.gml

```
button_type = "";

hovering = false;
clicked = false;

sprite_still = spr_button;
sprite_hover = spr_inverted_spr;
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_button);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_color(c_white);


//draw_text(x, y, text);
draw_text_ext(x, y, text, 20, 175);
```

LeftPressed.gml

```
sprite_index = sprite_still;
clicked = true;
```

LeftReleased.gml

```
sprite_index = sprite_hover;

if (hovering && clicked)
{
	clicked = false;

	switch(button_type)
	{
	  case "select_level":
	      room_goto(rm_level_select);
	      break;
	  case "new_game":
	              with(obj_game_controller)
	              {
	          new_game();
	      }
	      room_goto(rm_level_select);
	      break;
	       case "load_game":
	              if (file_exists("savedata.sav"))
	              {
	                     with(obj_game_controller)
```

```
                        {
                                load_game();
                        }
                }
            room_goto(rm_level_select);
            break;
    }
}
else
{
       clicked = false;
}
```

MouseEnter.gml

```
sprite_index = sprite_hover;
hovering = true;
```

MouseLeave.gml

```
sprite_index = sprite_still;
hovering = false;
clicked = false;
```

obj_start_controller

Create.gml

```
// Create three buttons using existing button object
newGameButton = instance_create_layer(room_width/2 - 200, 625, "UILayer",
obj_start_button);
newGameButton.image_xscale = 1.152381;
newGameButton.image_yscale = 1.316832;
newGameButton.text = "New Game";
newGameButton.button_type = "new_game";

loadGameButton = instance_create_layer(room_width/2 + 200, 625, "UILayer",
obj_start_button);
loadGameButton.image_xscale = 1.152381;
loadGameButton.image_yscale = 1.316832;
loadGameButton.text = "Load Game";
loadGameButton.button_type = "load_game";
// Only enable load button if save file exists
loadGameButton.enabled = file_exists("savedata.sav");
```

obj_detail_tutorial

Create.gml

```
depth = -10000;

//Tutorial Level Detailed Content
//Tutorial Level Encounter 1
board_content = @"Before every section, you are able to click on the chalkboard to
extend it.
```

```
Doing this will extend it, much like this, but will show you information to help you
with the upcoming challenges.

The information is based on Dr.Redfield's 'C Programming Workbook'.
Each section during your journey is based on each chapter with the relevant name.
So, every level will contain 3 sections which equal 112 total sections.
To get the full experience and become a coding master, please make sure to complete
the coding challenges at the end of each chapter on your own device using the
recommended IDE (Dev C++).

To get back to the game, click on the red arrow in the top right corner.";
```

DrawGUI.gml

```
draw_self();

draw_set_font(fnt_normal);
draw_set_halign(fa_left);
draw_set_valign(fa_top);
draw_set_color(c_white);

//draw_text_ext(60, 70, board_content_text, 20, 1130);
draw_text_ext(60, 70, board_content, 20, 1130);
```

obj_tutorial_controller

Create.gml

```
//Create Event for obj_tutorial_controller
global.can_move = false;
panel = instance_create_layer(640, 75, "UILayer", obj_tutorial_panel);
```

obj_ tutorial_encounter

Create.gml

```
//Create Event for obj_tutorial_encounter
global.can_move = false;
tutorial_question = true;

panel = instance_create_layer(640, 75, "UILayer", obj_tutorial_panel);
panel.tutorial_stage = 5;
//panel.text_current = 9;
```

Step.gml

```
if(!instance_exists(panel) && tutorial_question)
{
	encounter = instance_create_layer(640, 75, "UILayer",
obj_encounter_controller);
	tutorial_question = false;
}
```

obj_ tutorial_encounter

Create.gml

```
hovering = false;
clicked = false;
expanded = false;

show_debug_message("Depth: " + string(depth));


tutorial_stage = 0;  // Track which message to show

has_moved_right = false;   // Track if player has moved right
has_moved_left = false;   // Track if player has moved left
has_jumped = false;  // Track if player has jumped


depth = -100;
// Tutorial Stage 0 (Click)
text[0] = "Welcome to the world of coding! \n(Click to continue)";
text[1] = "Your name is Alan.\n You have been created to save our world from
destruction. \n(Click to continue)";
text[2] = "A dangerous bug invasion threatens to corrupt all programming
knowledge.\n(Click to continue)";
text[3] = "To stop this, you must master the art of coding to defeat these bugs and
save the digital world from collapse! \n(Click to continue)";

// Tutorial Stage 1 (Click)
text[4] = "Let's get you moving! \nUse the RIGHT ARROW KEY to move Alan to the right
throughout this screen.\n";
text[5] = "Now, use the LEFT ARROW KEY to move Alan left.\n";
text[6] = "Now press the SPACE BAR to jump. This will be how to defeat the bugs later
on!";

// Tutorial Stage 2 (Click)
text[7] = "Amazing! \nNow on to what you will need to know to defeat the bugs.\nClick
me now.";

// Tutorial Stage 3 (Click)
text[8] = "Now lets clear some bugs! \nMove all the way to the right of the screen to
go the next room.";

// Tutorial Stage 5 (Battle)
text[9] = "This is where you encounter the questions and the bugs.\n(Click to
continue)";
text[10] = "There are 2 phases with each encounter that you will need to complete so
that you can save the world. \n(Click to continue)";
text[11] = "In the first phase, you will be shown a question and have to select the
best option from 3 answer choices. Each level will have multiple questions based on
the section/chapter. \n(Click to continue)";
text[12] = "You can not continue until you answer each question correctly. \nHint:
The questions change every time so read them carefully! \n(Click to continue)";
```

```
text[13] = "<---------- These are your hearts.\n Everytime that you answer a
questions incorrectly, you will lose a heart.\n However, you can gain your heart back
if you answer it correctly. \n(Click to continue)";
text[14] = "Once you answer correctly, you will you will be able to move.\nThe bug in
the bottom right of the screen will also move. \n(Click to continue)";
text[15] = "Your mission is to stomp the bugs by jumping on them. However, the bug
can take a heart away if it hits you. \nTip: You also get 2 seconds of invincibility
once hit \n(Click to continue)";
text[16] = "At any time if you have 0 hearts, you will have be redirected to the
level selection screen where you will have to redo the level. \n(Click to continue)";
text[17] = "If you answer the questions and defeat the enemy, move right to continue.
Let's test this now by seeing if you were paying attention! \n(Click to continue)";


text_current = 0;
text_last = 1;
text_width = 900;


char_current = 1;
char_speed = 0.50;

//board_text = board[0, 0];

text[text_current] = string_wrap(text[text_current], text_width);
```

DrawGUI.gml

```
draw_self();
draw_set_font(fnt_question);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_colour(c_white);

var _len = string_length(text[text_current]);

if (char_current < _len)
{
    char_current += char_speed;
}

var _str = string_copy(text[text_current], 1, char_current);

// Clicking Events
if (tutorial_stage == 0 || tutorial_stage == 2 || tutorial_stage == 5)
{
	if (hovering == true)
	{
		draw_set_color(c_yellow);
		draw_text(room_width/2, y,  _str);
	}
	else
	{
		draw_set_color(c_white);
	}
```

```
}

if(tutorial_stage == 5 && text_current < 9)
{
	text_current = 9;
}

function unlock_room()
{
	var _len = string_length(text[text_current]);

	if (char_current < _len)
	{
		char_current = _len;
	}
	else
	{
		text[text_current] = string_wrap(text[text_current], text_width);
		char_current = 0;
	}

	global.tutorial_lock = false;
}


draw_text(room_width/2, y,  _str);
```

LeftPressed.gml

```
if(tutorial_stage == 0 || tutorial_stage == 2 || tutorial_stage == 5)
{
	clicked = true;
}
```

LeftReleased.gml

```
if(tutorial_stage == 0 || tutorial_stage == 2 || tutorial_stage == 5)
{
	if (hovering && clicked)
	{
		if(tutorial_stage == 0)
		{
			var _len = string_length(text[text_current]);

			if (char_current < _len)
		    {
				char_current = _len;
		    }
			else
		    {
				text_current++;

				if (text_current == 4)
		        {
```

```
                                text[text_current] = string_wrap(text[text_current],
text_width);
                                char_current = 0;
                                tutorial_stage++;
                                global.can_move = true;
                          }
                                else
                          {
                                text[text_current] = string_wrap(text[text_current],
text_width);
                                char_current = 0;
                          }
                    }
              }
              if(tutorial_stage == 2)
              {
                    if (!expanded)
                    {
                          clicked = false;
                          // Create dark overlay first (so it's behind the detail
panel)
                          instance_create_layer(0, 0, "UILayer", obj_dark_overlay);

                          extendedView = instance_create_layer(room_width/2,
room_height/2, "UILayer", obj_detail_tutorial);
                          extendedView.image_yscale = 4.5;

                          backButton = instance_create_layer(1190, 80, "UILayer",
obj_back_button);
                          backButton.backButton_type = "close_tutorial";

                          expanded = true;

                          visible = false;
                          global.can_move = false;
                    }
                    else
                    {
                          expanded = false;
                          visible = true;
                          global.can_move = true;
                          text_current++;
                          tutorial_stage++;
                          unlock_room();
                    }
              }
              if(tutorial_stage == 5)
              {
                    depth = -100;
                    show_debug_message("Object depth: " +
string(obj_heart_controller.depth));
                    var _len = string_length(text[text_current]);

                    if (char_current < _len)
                {
```

```
                        char_current = _len;
                }
                   else
                {
                        text_current++;

                        if (text_current == 18)
                    {
                            draw_set_color(c_white)
                            instance_destroy();
                    }
                        else if (text_current == 13)
                        {
                            depth = 100;
                        }
                        else
                    {
                            text[text_current] = string_wrap(text[text_current],
1200);
                            char_current = 0;
                    }
                }
            }
      }
}
else
{
      clicked = false;

}
```

MouseEntered.gml

```
if(tutorial_stage == 0 || tutorial_stage == 2 || tutorial_stage == 5)
{
      hovering = true;
}
```

MouseLeave.gml

```
if(tutorial_stage == 0 || tutorial_stage == 2 || tutorial_stage == 5)
{
      hovering = false;
      clicked = false;
}

//draw_text(room_width/2, y,  _str);
draw_set_color(c_white)
```

KeyDownSpace.gml

```
if (has_moved_right && has_moved_left && !has_jumped && tutorial_stage == 1)
{
      text_current++;
```

```
	text[text_current] = string_wrap(text[text_current], text_width);
	char_current = 0;
	tutorial_stage++;

	has_jumped = true;
}
```

KeyDownLeft.gml

```
if (has_moved_right && !has_moved_left  && tutorial_stage == 1)
{
	text_current++;

	text[text_current] = string_wrap(text[text_current], text_width);
	char_current = 0;

	has_moved_left = true;
}
```

KeyDownRight.gml

```
if(!has_moved_right && global.can_move  && tutorial_stage == 1)
{
	var _len = string_length(text[text_current]);

	  if (char_current < _len)
	{
		char_current = _len;
	}
	  else
	{
		text_current++;

		text[text_current] = string_wrap(text[text_current], text_width);
		char_current = 0;

		has_moved_right = true;
	}
}
```

obj_back_button

Create.gml

```
// obj_back_button Create Event
depth = -10001;
backButton_type = "";
hovering = false;
clicked = false;
```

DrawGUI.gml

```
draw_self();
```

LeftPressed.gml

```
// Left Pressed Event
image_alpha = 1;
clicked = true;
```

LeftReleased.gml

```
image_alpha = 0.5;

if (hovering && clicked)
{
	clicked = false;

	switch(backButton_type)
	{
		case "start":
			room_goto(rm_start);
			break;
		case "close_detail":
			instance_destroy(obj_dark_overlay);
			instance_destroy(obj_detail_panel);
			instance_destroy();
		    break;
		case "close_tutorial":
			instance_destroy(obj_dark_overlay);
			instance_destroy(obj_detail_tutorial);
			instance_destroy();
		    break;
		case "level":
			room_goto(rm_level_select);
			break;
	}
}
else
{
	clicked = false;
}
```

MouseEnter.gml

```
// Mouse Enter Event
image_alpha = 0.5;
hovering = true;
```

MouseLeave.gml

```
// Mouse Leave Event
image_alpha = 1;
hovering = false;
clicked = false;
```

obj_game_controller

Create.gml

```
// Create Event of obj_game_controller
// Level information - What sections are in each level
level_sections[0] = ["Introduction and Main", "Variables and Expressions", "Input and
Operations"];
level_sections[1] = ["Conditionals", "Functions", "Loops"];
level_sections[2] = ["Arrays", "Strings - Character Arrays", "Addresses and
Pointers"];
level_sections[3] = ["Enumerate and Structures", "File Input and Output", "Object
Oriented Programming"];

// Initialize global variables
// Load saved highest level if it exists
function load_game()//if (file_exists("savedata.sav"))
{
    var _buffer = buffer_load("savedata.sav");
    var _string = buffer_read(_buffer, buffer_string);
    buffer_delete(_buffer);

    var _loadData = json_parse(_string);
    global.highest_level = _loadData[$ "highest_level"] ?? 0;  // 0 = tutorial only
       global.game_complete = _loadData[$ "game_complete"] ?? false;
       show_debug_message("Highest level: " + string(global.highest_level));
       show_debug_message("Game Complete: " + string(global.game_complete));
}

function new_game()
{
    // Set default if no save exists
    globalvar highest_level;
       globalvar game_complete;
    global.highest_level = 0;  // Start with only tutorial unlocked
       global.game_complete = false;

       show_debug_message("Highest level: " + string(global.highest_level));
       show_debug_message("Game Complete: " + string(global.game_complete));
}

// Initialize other variables (not saved)
globalvar can_move, current_level; //current_encounter,
//global.current_encounter = 1;  // Always start at first encounter
global.current_level = 0;  // 0=tutorial, 1=C, 2=C++, 3=Python, 4=Java
can_move = true;  // Will be set to false in encounter rooms

// Which section within level (0-2) and Which question we're on in current section
globalvar current_section, question_in_section;
global.current_section = 0;      // Start with first section
global.question_in_section = 0;  // Start with first question in section

//Store number of questions in section
globalvar number_of_questions;
global.number_of_questions = 0;

globalvar tutorial_lock;
global.tutorial_lock = true;
```

```
// Create a save function
function save_game()
{
    var _saveData = {
             highest_level: global.highest_level,
             game_complete: global.game_complete
             };

    var _string = json_stringify(_saveData);
    var _buffer = buffer_create(string_byte_length(_string) + 1, buffer_fixed, 1);
    buffer_write(_buffer, buffer_string, _string);
    buffer_save(_buffer, "savedata.sav");
    buffer_delete(_buffer);

      show_debug_message("Saving game... Highest level: " +
string(global.highest_level));
      show_debug_message("Game Complete: " + string(global.game_complete));
    show_debug_message("Save location: " + working_directory + "savedata.sav");
}
```

obj_heart_controller

Create.gml

```
// Create Event of obj_heart_controller
max_hearts = 5;
current_hearts = max_hearts;
heart_spacing = 40;  // Space between hearts
start_x = 65;        // Starting x position for first heart 32
start_y = 50;        // Y position for hearts 192

flash_hearts = false;
flash_timer = 0;
```

Step.gml

```
// Step Event
if (current_hearts <= 0)
{
    // Game Over
    room_goto(rm_start);
    current_hearts = max_hearts;  // Only reset hearts on game over
}

if(flash_hearts)
{
    flash_timer++;
    if(flash_timer > 30)
      {  // Half second flash
        flash_hearts = false;
        flash_timer = 0;
    }
}
```

DrawGUI.gml

```
if (room != rm_start && room != rm_level_select && room != rm_game_over && room != 
rm_level_complete && room != rm_tutorial_learn && room != rm_achievements) // Add any 
other rooms where hearts shouldn't show
{
	for(var i = 0; i < max_hearts; i++)
	{
	    if(i < current_hearts)
	       {
	        if(flash_hearts && flash_timer mod 6 < 3)
	             {
	            draw_sprite_ext(spr_heart, 0, start_x + (i * heart_spacing), 
start_y, 1, 1, 0, c_red, 1);
	        }
	        else
	             {
	            draw_sprite(spr_heart, 0, start_x + (i * heart_spacing), start_y);
	        }
	    }
	    else
	       {
	        draw_sprite(spr_heart, 1, start_x + (i * heart_spacing), start_y);
	    }
	}
}
```