Owen Scruggs

March 25, 2025

Prof. Macalintal

ENC1102

Conventions in RPG Maker VX Ace Scripting: What Makes an Ideal Script?

Owen Scruggs

Reflective Cover Letter

The purpose of this paper is to investigate what constitutes an ideal script layout for RPG Maker VX Ace by synthesizing perspectives from practicing scripters and analyzing real scripts through rhetorical code studies. This work is designed to speak directly to hobbyist and independent game developers, particularly those who rely on shared codebases without formal training in programming. My goal was to surface the often-overlooked rhetorical decisions that go into script readability, structure, and accessibility. Recognizing my audience as both academic and creative—those studying digital literacy and those immersed in making—I was careful to balance analytical rigor with grounded examples from real-world practice. This dual audience shaped how I framed my inquiry and how I presented my findings with clarity and precision.

My research process involved narrowing my focus from a general interest in RPG Maker VX Ace to a specific inquiry into script layout, which emerged organically from early brainstorming and my annotated bibliography. Conducting interviews with participants at different experience levels helped me define and refine my codes, and analyzing three contrasting scripts grounded my research in real practices. I revised heavily throughout: my research question evolved with each coding cycle, and I continually clarified my writing to reflect more precise observations. One major challenge was avoiding assumptions in interpreting participant feedback—revisiting the transcripts helped me stay faithful to what was actually said. Each draft of my outline and each coding pass deepened my insight and clarified my structure. Ultimately, organizing my results section by script and participant allowed me to spotlight nuanced trends more effectively than a generalized summary would have.

This project demonstrates progress across several course outcomes. I developed a meaningful line of inquiry into how digital literacy manifests in community-based programming environments (Generating Inquiry). I incorporated both interviews and script samples (Multiple Ways of Writing) and closely evaluated all primary and secondary sources for relevance and credibility (Information Literacy). The final paper itself is a genre-aware piece of academic writing that contributes to scholarly conversations on code literacy and rhetorical practice (Research Genre Production & Contributing Knowledge). And finally, my work reflects extensive revision in both content and structure, using feedback and self-assessment to guide each stage (Revision). This project challenged me to think of code not just as functional, but as communicative—and to honor the literacy practices of a vibrant, if often overlooked, creative community.

Introduction:

In the world of RPG Maker VX Ace development, scripting is more than just functional coding—it is a form of communication. As game creators customize their projects through Ruby-based RGSS3 scripts, the clarity, structure, and purpose of these scripts become critical for both usability and collaboration. Scholars like Rieder and Peppler (2021) note that the organization of code reflects not just technical intention but social and rhetorical values. Similarly, scholars such as Fiadotau (2022) and García (2018) emphasize the importance of readable, accessible formats for knowledge sharing in fan and modding communities. Across studies of digital literacy and participatory cultures, the act of coding is increasingly recognized as a literate, meaning-making practice that mirrors the conventions of more traditional academic writing (Knutson, 2020; Sheridan et al., 2020). However, while these works emphasize how software users communicate meaning, few have explored how genre conventions of layout affect the scripts themselves in tools like RPG Maker VX Ace, especially from the perspective of everyday scripters rather than professional developers.

Although there is a growing body of research analyzing code rhetorics, much of this scholarship centers on broader sociotechnical systems (Fiadotau, 2022) or educational programming environments (Knutson, 2020; García, 2018), often emphasizing how code is taught or functions within institutions. These contexts tend to overlook the informal, everyday practices of organizing and presenting code used by hobbyists or fan creators, such as those using RPG Maker VX Ace. For example, while Knutson explores how young programmers construct meaning through Scratch in classroom settings, and Fiadotau discusses community values in modding cultures, there remains a blind spot regarding how individual scripters choose to visually and structurally arrange code to communicate intentions, logic, and purpose within small-scale, user-driven development tools.

To explore these questions, this study relies on interviews with scripters and close rhetorical analysis of user-created scripts to identify patterns in how layout is used rhetorically. These methods allowed me to uncover shared expectations, tensions, and variations in what scripters believe makes a script usable or communicative. The findings illuminate a deeper understanding of how genre awareness and rhetorical intent influence code structure—even in hobbyist spaces. The following section outlines how my data was collected and analyzed to support this investigation.

Methods:

To investigate what constitutes a clear and ideal script layout for RPG Maker VX Ace from the perspective of practicing scripters, I used a qualitative, mixed-method approach grounded in rhetorical code studies. My primary data sources consisted of three structured interviews and three example scripts, each selected to reflect a range of styles and complexities. I analyzed these sources through an iterative process of thematic coding, informed by rhetorical theory and genre analysis frameworks from scholars such as DeVoss et al. (2020) and Potts (2014), to identify consistent features and expectations regarding how scripts should be structured and annotated for clarity and reusability.

Participants and Interviews

My interviews targeted scripters with varying levels of experience in RPG Maker VX Ace, chosen to reflect the diversity of perspectives within the scripting community. The three interviewees were:

- Interviewee 1, a beginner who has minimal experience reading or writing scripts and is still learning how the RGSS3 language works.
- Interviewee 2, an intermediate scripter with moderate familiarity with the engine's scripting environment. While able to read and adapt scripts, they are not yet fluent in writing complete, original systems.
- Interviewee 3, an advanced user who fully understands RGSS3, writes original scripts from scratch, and is capable of modifying complex systems. However, Interviewee 3 does not produce commercial scripts and participates mainly in personal or hobbyist projects.

Each participant was shown three scripts and asked to describe what they understood about the script's layout, clarity, and overall usability. None of the participants were shown scripts they had authored. While some recognized specific scripts—namely, a Japanese-language script focused on evasion mechanics (Script 2) and a well-known script by Yanfly (Script 3)—their insights were based on reader familiarity rather than authorship.

The interviews were conducted in a semi-structured format via digital messaging platforms, such as Discord, with consent obtained beforehand. The interviews were then transcribed and coded for recurring themes, including accessibility, clarity, modularity, and narrative commentary (in-line documentation that "tells the story" of the code).

Script Sample Selection

The three scripts selected for analysis were chosen for their diversity in language, layout style, and authorial context:

- Script 1 is a fully custom, English-language script written by an anonymous developer. It features minimal comments and appears to be written for personal use or prototyping, making it ideal for analyzing unmediated scripting style.
- Script 2 is a translated Japanese script by Hoshigata, known for dense but modular formatting and complex logic structures. This script was selected due to its widespread recognition in the RPG Maker VX Ace community and its rich example of culturally different commenting styles.
- Script 3 is a large, well-documented plugin from Yanfly, whose works are considered best-in-class for accessibility and structure. It was chosen to represent a high benchmark in terms of script layout conventions.

These scripts were presented to each interviewee in full, without any identifying information or author attribution. Participants were encouraged to describe what they could or couldn't understand and what helped or hindered their comprehension of the code.

Coding and Analysis

I used open and axial coding to analyze both the interview transcripts and the scripts themselves. My initial codes were drawn from rhetorical code studies literature, particularly the work of DeVoss et al. (2020), who emphasized that code should be seen as a form of communication, and Potts (2014), who argued for the importance of audience-aware design in technical documents. From these foundations, I developed a coding schema focused on the following categories:

- Modularity: How easily the script can be expanded, reused, or altered.
- Debugging: Presence of built-in error handling or failure messages.
- Documentation: Quantity and clarity of in-line comments, warnings, and instructions.
- Clarity vs. Functionality: Whether the script favors readability or is focused solely on working as intended, even at the cost of clarity.
- Error Responsibility: Who is expected to catch and fix problems—the script author or the user?
- Translation & Accessibility: Whether the script assumes knowledge of English or includes obstacles to non-fluent readers.

Each interview was independently coded according to this schema, and excerpts were highlighted and tagged with corresponding codes. For example, a participant describing difficulty understanding a script due to "all the logic being lumped together in one huge block" would be tagged with Clarity and Functionality. The same schema was used to annotate the three sample scripts, allowing for cross-comparison between participant impressions and observable script features.

This coding allowed me to triangulate the participant feedback and textual features in the scripts themselves. The codes that emerged most frequently and powerfully are highlighted in the results section, where I describe what I learned from each piece of primary data before transitioning to a broader discussion of these findings in context.

Results:

This section presents my coded analysis of the three selected scripts, as interpreted by participants of varying scripting experience levels. Each script is analyzed individually, with participant responses categorized according to the six major codes derived from my rhetorical framework: Modularity, Debugging, Documentation, Clarity vs. Functionality, Error Responsibility, and Translation & Accessibility. Rather than presenting full interview transcripts or script excerpts here, I focus on relevant segments of dialogue and code features; full documents are available in the appendices.

Script 1: Custom English Script (Anonymous)

Overview:

Script 1 was written in English and contains virtually no in-line comments. The script appears to be functional but lacks documentation or clear segmentation. It was presented as an example of an unmediated personal scripting style.

Modularity

- Interviewee 3 identified a lack of modular structure, noting that the script used "long method blocks without breakpoints," making future edits difficult. They suggested the script was likely intended for personal use only.
- Interviewee 2 said they "couldn't tell where things started or ended," especially in the update loop. This indicates the absence of a modular layout hindered navigation.
- Interviewee 1, with limited experience, said they "couldn't figure out what any part of it did," reflecting how non-modular code excludes novices from engaging with the script at all.

Debugging

- None of the participants identified any built-in error handling.
- Interviewee 3 pointed out that "if anything broke, there'd be no indication," indicating the absence of debugging support.

Documentation

- All three participants noted the absence of comments.
- Interviewee 2 stated, "Even just a comment saying what the class is supposed to do would help."
- Interviewee 1 suggested "definitely comments for the first part—to keep track of what each script is attempting to do and how they interact."

Clarity vs. Functionality

- Interviewee 3 speculated the script "probably works fine," but emphasized that the structure prioritizes functionality over clarity.
- Interviewee 1 expressed confusion, saying "I have no idea what I'm looking at," reinforcing that scripts focused solely on functionality are inaccessible to learners.

Error Responsibility

- The script places all responsibility on the user to identify and fix problems.
- Interviewee 2 noted that "any mistakes would crash the game without telling you why," highlighting the lack of safety nets.

Translation & Accessibility

- Since the script is in English and designed by an English speaker, translation was not an issue.
- However, the absence of documentation still made it inaccessible, particularly for beginners like Interviewee 1.

Script 2: Translated Japanese Script (Hoshigata)

Overview:

Script 2 is a Japanese-language plugin originally written by Hoshigata. While known for its functionality, this version of the script is untranslated, making it a case study in linguistic accessibility and structural clarity.

Modularity

- Interviewee 3 stated that Hoshigata's code is "usually very good at being modular," though they noted that this version was hard to assess without translation.
- Interviewee 2 said, "There's definitely a lot of structuring here... but I can't really tell how it links together."
- Interviewee 1 acknowledged that even though they recognized the script, "without translation... I'm kinda just looking at nothing."

Debugging

- No participant identified any visible debugging functions.
- Interviewee 1 argued that scripts should have some protection "within any bounds specified in comments," but accepted that users take on responsibility after editing.

Documentation

- Interviewee 3 commented that “Japanese scripts usually have tons of comments,” but said this is only helpful “if you read Japanese.”
- Interviewee 2 said, “It seems well commented, but none of it's in English.”
- Interviewee 1 reinforced this, saying “translating the intro will give a clear definition I’m sure... but without it I’m just looking at nothing.”

Clarity vs. Functionality

- Interviewee 3 observed that despite the language barrier, the script’s layout looked “well-structured,” and said this was likely a case of clarity for native readers, not outsiders.
- Interviewee 2 noted that, while the layout seemed intentional, “it doesn’t help if you can’t read any of it.”
- Interviewee 1 pointed out that some parts like “absolute hit” were self-explanatory, but admitted that otherwise, “I can’t understand it.”

Error Responsibility

- Interviewee 3 observed that the script didn’t include visible fail-safes but noted that this is “common in Japanese scripts—they expect the user to install it properly.”
- Interviewee 1 echoed this sentiment: “If the script is making errors from being imported, that’s a problem.” But once edited, “the user should be responsible.”

Translation & Accessibility

- All participants identified language as the main barrier.

- Interviewee 1 remarked, “If I can’t translate it, then I can’t use it.”
- Interviewee 2 said they had used a translated version before, but in this form, “it’s not usable.”
- Interviewee 3 emphasized the importance of translation, calling untranslated scripts “a huge obstacle to learning from them.”

Script 3: Yanfly’s Ace Shop Options (Well-Documented Plugin)

Overview:

Script 3 is a highly documented, modular plugin created by Yanfly, a well-known figure in the RPG Maker VX Ace community. The script includes header instructions, organized sections, and consistent in-line commenting. It is recognized for balancing readability and functionality and is often held up as a model for accessible scripting.

Modularity

- Interviewee 3 noted that the script “uses sections really well,” allowing for easy extension or customization.
- Interviewee 2 appreciated how it was “easy to see where changes could be made,” indicating a high degree of modularity.
- Interviewee 1, despite being a beginner, said “I can at least tell what the different sections are,” showing that the modular format supported understanding at all experience levels.

Debugging

- Interviewee 3 pointed out the inclusion of “clear method separation and scene isolation,” which helps prevent broad failures across unrelated systems.

- Interviewee 2 noted that if something breaks, "it's likely to be in a localized section," making debugging easier.

Documentation

- All three participants praised the script's detailed commentary.
- Interviewee 1 said, "This one feels like it's holding my hand the whole way."
- Interviewee 2 referred to it as "beginner-friendly" because of the section headers and inline notes.
- Interviewee 3 described the documentation as "clear and consistent," adding that it "explains how it's working without being overwhelming."

Clarity vs. Functionality

- Interviewee 3 observed that Yanfly "writes with the user in mind," noting a rare balance between code clarity and powerful functionality.
- Interviewee 1 described the script as "understandable even if I don't know what every part does," showing how clear layout helped bridge knowledge gaps.
- Interviewee 2 said they "didn't feel intimidated" by the script, unlike the others, indicating that its clarity supported approachability.

Error Responsibility

- Interviewee 1 felt that this script "helps prevent errors by explaining everything upfront."
- Interviewee 3 emphasized that the structure "minimizes the chance of user mistakes," showing that the author assumed responsibility for guiding proper use.

- Interviewee 2 noted that “if I made a mistake, I could probably fix it,” reflecting confidence inspired by the clarity of the code.

Translation & Accessibility

- Since the script is written in English with ample explanation, accessibility was not an issue for any of the participants.
- Interviewee 1 explicitly called this script “the easiest to understand,” citing both language and organization as key factors.
- Interviewee 3 stated that Yanfly’s scripts are “some of the most accessible out there,” a sentiment echoed by Interviewee 2, who had used translated scripts before but preferred this level of readability.

Discussion:

This study set out to answer the question: *What do practicing scripters consider an ideal script layout in RPG Maker VX Ace, and how do issues like documentation, modularity, and accessibility factor into those preferences?* My primary data—three interview transcripts and three scripts of varying complexity and clarity—revealed a surprisingly consistent set of values across skill levels. While the interviews highlighted different pain points depending on experience, all participants gravitated toward similar ideals: clear documentation, structured modularity, and accessible language. These findings suggest that while scripting ability varies, expectations around script readability and usability are broadly shared across the community.

Revisiting the Scholarly Framework

DeVoss et al. (2020) argue that code is not just functional—it is rhetorical. Code communicates to both machines and human readers, and as such, must be constructed with audience awareness. Similarly, Potts (2014) emphasizes the ethical stakes of technical writing, particularly in open-source or collaborative environments. My data affirms both perspectives: scripts that failed to anticipate the reader's needs (e.g., Script 1) alienated users across experience levels, while scripts designed with documentation and structure in mind (e.g., Script 3) enabled comprehension and confidence even among beginners.

This intersection between functionality and rhetorical design was most visible in how participants responded to Documentation and Modularity. Yanfly's script, which excelled in both, was seen as "holding [the reader's] hand" (Interviewee 1) and "easy to change without breaking stuff" (Interviewee 3). In contrast, the undocumented custom script was called "like trying to read a foreign language with no translator" (Interviewee 1) and "totally unclear where it starts or ends" (Interviewee 2). Even Script 2, though functionally impressive, suffered from its assumption of fluency and cultural familiarity, reinforcing Potts' (2014) concern about access barriers in digital texts.

Central Codes and Their Broader Implications

While all six codes appeared throughout my data, three stood out as most impactful across skill levels: Documentation, Modularity, and Translation & Accessibility.

- Documentation emerged as the baseline requirement for any degree of understanding. Without it, even advanced users had to reverse-engineer functionality. As Interviewee 3 noted, "I shouldn't have to read the whole file to figure out what it's doing." Good documentation, meanwhile, empowered even novices to participate, supporting DeVoss et al.'s idea of code as inclusive communication.
- Modularity wasn't just a technical preference—it was a literacy scaffold. Scripts broken into readable chunks helped learners navigate code and helped experts modify it without unintended consequences. Interviewee 3's praise for "well-separated scenes and method groups" in Script 3 highlighted how modular design supports both reuse and learning.
- Translation & Accessibility became a point of divergence, particularly in Script 2. While Interviewee 2 and Interviewee 3 could parse the logic due to prior experience, Interviewee 1's reaction—"I don't know what it's saying or what it does"—illustrated how language barriers and cultural differences can render otherwise usable scripts inaccessible. This reinforces the rhetorical dimension of code and the need for inclusive practices in script design.

Interestingly, Error Responsibility was only discussed when absent. Scripts that lacked error-handling (like Script 1) were assumed to break silently or crash, pushing all troubleshooting onto the user. Conversely, well-structured scripts like Yanfly’s minimized the risk of user error through clarity alone, suggesting that readers associate responsibility with rhetorical design as much as with technical safeguards.

Conclusion

This study demonstrates that scripters across experience levels share common expectations about what makes a script usable, readable, and reusable in RPG Maker VX Ace. Despite differences in technical ability, all participants emphasized the value of clear documentation, modular structure, and accessible language. Scripts that failed to meet these criteria, such as the undocumented custom script or the untranslated Japanese script, created significant barriers—even for advanced users. In contrast, Yanfly's highly structured and well-commented script served as a model for how thoughtful script layout can facilitate both comprehension and adaptation.

These findings reinforce the rhetorical nature of code. A script is not just a functional tool—it is a text designed for human readers. As DeVoss et al. (2020) and Potts (2014) suggest, technical writing—including programming—carries ethical and communicative responsibilities. Scripts that anticipate their audience through documentation and structure do more than work well—they teach, invite collaboration, and lower the barrier to entry for others.

At the same time, this study is not without limitations. All interviewees were English-speaking, and while one script included Japanese content, I was not able to include participants with Japanese fluency or perspectives from non-English scripting communities. Future research could explore how layout conventions differ across linguistic and cultural contexts or how scripting literacy develops in community forums and modding spaces. Additionally, while my sample captured a range of experience levels, it was limited to three participants. A broader set of interviews could yield deeper insights into how factors like education, professional coding background, or game development goals shape layout preferences.

Nevertheless, this study contributes a meaningful step toward understanding how code functions not only as a tool but as a rhetorical object. By foregrounding the perspectives of everyday users and creators within RPG Maker VX Ace, I highlight the need for scripting practices that are not just technically sound—but also human-readable.

Works Cited

Brock, André. *Distributed Blackness: African American Cybercultures*. NYU Press, 2020.

Clarke, Rachel Ivy, et al. "Making Mad Scientists: The Role of RPG Maker in Creating STEAM Learning Spaces." *International Journal of Designs for Learning*, vol. 9, no. 1, 2018, pp. 130–144.

DeVoss, Dànielle Nicole, et al. *Making Space: Writing Instruction, Infrastructure, and Multiliteracies*. University of Michigan Press, 2020.

Fiadotau, Mikhail. "Hacking the Toolbox: Game Engines and Metagame Rhetorics." *First Person Scholar*, 2016. https://www.firstpersonscholar.com/hacking-the-toolbox/

Fiadotau, Mikhail. "Games as Reflective Practice: How Game Making Can Support Student Learning." *E-Learning and Digital Media*, vol. 16, no. 3, 2019, pp. 178–191.

Schatten, Markus, et al. "A Survey on Game Development Using the Unity Game Engine." *Tem Journal*, vol. 9, no. 4, 2020, pp. 1552–1559.

Owens, Trevor. "Modding the Historians' Code: Building an Institute for Advanced Topics in Digital History." *Journal of Digital Humanities*, vol. 1, no. 1, 2011.

Appendices

Color codes:

Modularity, Debugging, Documentation, Clarity vs. Functionality, Error Responsibility, Translation and Accessibility

Scripts:

Script 1:

```
class ConsoleTextEffect
  attr_reader :running

  def initialize(viewport, x, y, width, height, font_size = 20, line_spacing = 4)
    **puts "[DEBUG] Initializing ConsoleTextEffect"**  # (Debugging)
    ...
    @sprite.z = 200
    **create_cursor**  # (Modularity: delegated into its own method)
    reset
  end

  def create_cursor
    **puts "[DEBUG] Creating cursor sprite"**  # (Debugging)
    ...
    **puts "[DEBUG] Cursor sprite created: #{@cursor_sprite}"**  # (Debugging)
  end

  def reset
    return if disposed?
    **puts "[DEBUG] Resetting ConsoleTextEffect"**  # (Debugging)
    ...
    **create_cursor unless @cursor_sprite**  # (Modularity)
    reset_cursor
  end

  def start_text(text, speed = 2, wait_for_input = true, auto_erase = false)
    **puts "[DEBUG] Starting text: #{text.inspect}"**  # (Debugging)
    ...
  end
```

```
def update
  return if disposed? || !@running
  ...
  if @awaiting_input
    if @wait_for_input
      handle_input
    else
      handle_auto_erase if @auto_erase
      @running = false
    end
    return
  end
  ...
  process_current_line
end

def handle_input
  **if Input.trigger?(:C) || Input.trigger?(:B)**  # (Functionality over clarity)
    **puts "[DEBUG] Input received"**  # (Debugging)
    ...
  end
end

def handle_auto_erase
  **puts "[DEBUG] Handling auto-erase"**  # (Debugging)
  clear_text
  reset_cursor
end

def process_current_line
  ...
  else
    puts "[DEBUG] Awaiting input"  # (Debugging)
    @awaiting_input = true
  end
end

def clear_text
  return if disposed?
  **puts "[DEBUG] Clearing text"**  # (Debugging)
  ...
end

def dispose
  return if disposed?
  **puts "[DEBUG] Disposing ConsoleTextEffect"**  # (Debugging)
```

```
    ...
  end

  def dispose_cursor
    return unless @cursor_sprite
    **puts "[DEBUG] Disposing cursor sprite: #{@cursor_sprite}"**  # (Debugging)
    ...
    **puts "[DEBUG] Cursor sprite disposed."**  # (Debugging)
  end

  def disposed?
    @disposed == true
  end
end

class Scene_Base
  alias console_text_effect_update update
  def update
    ...
    **@children.reject!(&:disposed?)**  # (Modularity: managing children in a reusable way)
  end

  def add_child(child)
    **puts "[DEBUG] Adding child to Scene_Base: #{child}"**  # (Debugging)
    ...
  end

  def remove_child(child)
    **puts "[DEBUG] Removing child from Scene_Base: #{child}"**  # (Debugging)
  end

  def dispose_all_console_effects
    **puts "[DEBUG] Clearing all console effects in Scene_Base"**  # (Debugging)
    ...
    **puts "[DEBUG] Disposing console effect: #{child}"**  # (Debugging)
    ...
    **puts "[DEBUG] All console effects cleared."**  # (Debugging)
  end
end

class Game_Interpreter
  def start_console_text(...)
    **puts "[DEBUG] Starting console text"**  # (Debugging)
    ...
    **SceneManager.scene.dispose_all_console_effects if
SceneManager.scene.respond_to?(:dispose_all_console_effects)**
```

```
  # (Error Responsibility: shifts burden of cleanup to SceneManager)

  ...
  Fiber.yield while @console_effect.running  # (Functionality over clarity)
 end

 def end_console_text
  return unless @console_effect
  **puts "[DEBUG] Ending console text"**  # (Debugging)
  ...
 end

 def clear_console_text
  **puts "[DEBUG] clear_console_text called"**  # (Debugging)
 end

 def dispose_all_console_effects
  **puts "[DEBUG] Disposing all console effects from Game_Interpreter"**  # (Debugging)
  ...
 end

 private

 def initialize_or_reuse_console_effect(...)
  **puts "[DEBUG] Initializing or reusing console effect"**  # (Debugging)
  ...
 end

 def dispose_viewport
  return unless @viewport
  **puts "[DEBUG] Disposing viewport: #{@viewport}"**  # (Debugging)
 end
end
```

Script 2:

```
#==========================================================================
=====
# ■ RGSS3 絶対命中/絶対回避特徴＆アイテム/スキル Ver1.00　by 星潟
#------------------------------------------------------------------------
# 命中タイプ別、もしくは全てのアイテムについて
# 絶対命中/絶対回避化させる特徴を作成する事ができるようになります。
# また、絶対命中するアイテム/スキルの作成も可能になります。
# 命中/回避関連の全てのスクリプトよりも下に配置される事をお勧めします。
#===============================================================================
=====
**# ★設定例（アクター・エネミー・ステート・装備品のメモ欄に設定）** #
(Documentation – usage examples)
#-------------------------------------------------------------------------------
# <全絶対命中>
# このキャラクターによる全てのスキル/アイテムが絶対に命中します。
...
# <絶対命中>
# このスキル/アイテムは絶対に命中します。
#==================================================
============================

module CertaintyHit
  **#絶対命中と絶対回避が同時に計算される場合**
  **#絶対命中と絶対回避のどちらを優先するかを決定します。**  # (Documentation, Clarity vs.
Functionality)
  **#0の場合は命中を優先します。**
  **#1の場合は回避を優先します。**
  **#2の場合は絶対命中も絶対回避もなかったことにして本来の処理を行います。**

  Type  = 0  # (Modularity – user-configurable toggle)

  Words1 = ["AbsoluteHit","AbsolutePhys","AbsoluteMag","AbsoluteAll"]  # (Modularity)
  Words2 = ["必中絶対回避","物理絶対回避","魔法絶対回避","全絶対回避"]
  Word = "絶対命中"

  Value = [9.99, -9.99]  # (Functionality: hard-coded behavior overrides)

  def self.words(type)
    type ? Words1 : Words2
  end
end

class Game_Battler < Game_BattlerBase
  alias item_hit_certainty item_hit
  def item_hit(user, item)
    **#設定別に処理。**  # (Documentation)

    case CertaintyHit::Type
    when 0
      **return CertaintyHit::Value[0] if certainty_hit_execute(user, item)**  # (Functionality, Error
Responsibility: silent override of default logic)
      return CertaintyHit::Value[1] if certainty_eva_execute(user, item)
    when 1
      return CertaintyHit::Value[1] if certainty_eva_execute(user, item)
      return CertaintyHit::Value[0] if certainty_hit_execute(user, item)
    when 2
      return CertaintyHit::Value[0] if certainty_hit_execute(user, item) && !certainty_eva_execute(user, item)
      return CertaintyHit::Value[1] if certainty_eva_execute(user, item) && !certainty_hit_execute(user, item)
    end

    item_hit_certainty(user, item)  # (Clarity vs. Functionality – falls back to default method)
  end

  alias item_eva_certainty item_eva
  def item_eva(user, item)
    case CertaintyHit::Type
    when 0
      return CertaintyHit::Value[1] if certainty_hit_execute(user, item)
      return CertaintyHit::Value[0] if certainty_eva_execute(user, item)
    when 1
      return CertaintyHit::Value[0] if certainty_eva_execute(user, item)
      return CertaintyHit::Value[1] if certainty_hit_execute(user, item)
    when 2
      return CertaintyHit::Value[1] if certainty_hit_execute(user, item) && !certainty_eva_execute(user, item)
      return CertaintyHit::Value[0] if certainty_eva_execute(user, item) && !certainty_hit_execute(user, item)
    end
    item_eva_certainty(user, item)
  end

  def certainty_hit_execute(user, item)
    return true if item.certainty_hit_item
    return true if user.certainty_hit(3) or user.certainty_hit(item.hit_type)
    false
  end

  def certainty_eva_execute(user, item)
    **#味方へのスキルの場合は絶対回避は行わない。**  # (Documentation, Error Responsibility –
silent override of logic)
    return false if item.for_friend? && !item.for_opponent? && user.actor? == self.actor?
    return true if certainty_eva(3) or certainty_eva(item.hit_type)
    false
  end

  def certainty_hit(type)
    feature_objects.any? {|f| f.certainty_hit_array[type]}  # (Modularity – behavior changes based on database
tags)
  end
```

```
  def certainty_eva(type)
    feature_objects.any? {|f| f.certainty_eva_array[type]}  # (Modularity)
  end
end

class RPG::BaseItem
  def certainty_hit_array
    @certainty_hit_array ||= create_certainty_hit_eva_array(true)
  end

  def certainty_eva_array
    @certainty_eva_array ||= create_certainty_hit_eva_array(false)
  end

  def create_certainty_hit_eva_array(type)
    **CertaintyHit.words(type).inject([]) {|r,t| r.push(/<#{t}>/ =~ note)}**  # (Translation & Accessibility –
relies on note tags in Japanese or keywords)
  end
end

class RPG::UsableItem < RPG::BaseItem
  def certainty_hit_item
    (@certainty_hit_item ||= /<#{CertaintyHit::Word}>/ =~ note ? 1 : 0) == 1  # (Translation & Accessibility)
  end
end
```

Script 3:

```
#==============================================================================
=====
#
# ▼ Yanfly Engine Ace - Ace Shop Options v1.01
# -- Last Updated: 2012.01.05
# -- Level: Normal, Hard
# -- Requires: n/a
#
#==============================================================================
=====

$imported = {} if $imported.nil?
$imported["YEA-ShopOptions"] = true

#==============================================================================
=====
# ▼ Updates
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
# 2012.01.05 - Compatibility Update: Equip Dynamic Stats
# 2012.01.03 - Started Script and Finished.
#
#==============================================================================
=====
# ▼ Introduction
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
# The RPG Maker VX Ace shop scene is relatively basic. It provides adequate
# information, but not really enough to let the player know what they're
# actually buying or even selling. This script enables shops to show more than
# just the basic information displayed in RPG Maker VX Ace and even allow for
# custom commands to be inserted.
#
#==============================================================================
=====
# ▼ Instructions
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
# To install this script, open up your script editor and copy/paste this script
# to an open slot below ▼ Materials/素材 but above ▼ Main. Remember to save.
#
# -----------------------------------------------------------------------------
# Item Notetags - These notetags go in the item notebox in the database.
# -----------------------------------------------------------------------------
# <image: string>
# Uses a picture from Graphics\Pictures\ of your RPG Maker VX Ace Project's
# directory with the filename of "string" (without the extension) as the image
# picture shown in the Ace Shop Options.
#
# -----------------------------------------------------------------------------
# Weapon Notetags - These notetags go in the weapon notebox in the database.
# -----------------------------------------------------------------------------
# <image: string>
# Uses a picture from Graphics\Pictures\ of your RPG Maker VX Ace Project's
# directory with the filename of "string" (without the extension) as the image
# picture shown in the Ace Shop Options.
#
# -----------------------------------------------------------------------------
# Armour Notetags - These notetags go in the armour notebox in the database.
# -----------------------------------------------------------------------------
# <image: string>
# Uses a picture from Graphics\Pictures\ of your RPG Maker VX Ace Project's
# directory with the filename of "string" (without the extension) as the image
# picture shown in the Ace Shop Options.
#
#==============================================================================
=====
# ▼ Compatibility
# =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
# This script is made strictly for RPG Maker VX Ace. It is highly unlikely that
# it will run with RPG Maker VX without adjusting.
#
#==============================================================================
=====

module YEA
  module SHOP

    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    # - Shop Command Window Settings -
    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    # Here, you can adjust the order at which the commands appear (or even
    # remove commands as you see fit). Here's a list of which does what:
    #
    # -------------------------------------------------------------------------
    # :command         Description
    # -------------------------------------------------------------------------
    # :buy             Buys items from the shop. Default.
    # :sell            Sells items top the shop. Default.
    # :cancel          Leaves the shop. Default.
    #
    # :equip           Allows the player to change equipment inside the shop.
    #
    # :totorishop      Requires Kread-EX's Synthesis Shop.
    #
    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    COMMANDS =[
      :buy,          # Buys items from the shop. Default.
      :sell,         # Sells items top the shop. Default.
      :equip,        # Allows the player to change equipment inside the shop.
      :totorishop,   # Requires Kread-EX's Synthesis Shop.
      :cancel,       # Leaves the shop. Default.
    # :custom1,      # Custom Command 1.
    # :custom2,      # Custom Command 2.
    ] # Do not remove this.

    #--------------------------------------------------------------------------
    # - Shop Custom Commands -
    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    # For those who use scripts to that may produce unique effects for their
    # shops, use this hash to manage the custom commands for the Shop Command
    # Window. You can disable certain commands or prevent them from appearing
    # by using switches. If you don't wish to bind them to a switch, set the
    # proper switch to 0 for it to have no impact.
    #--------------------------------------------------------------------------
    CUSTOM_SHOP_COMMANDS ={
    # :command => ["Display Name", EnableSwitch, ShowSwitch, Handler Method],
      :equip   => [      "Equip",           0,         0, :command_equip],
      :totorishop => [ "Synthesis",          0,      0, :command_synthshop],
      :custom1 => [ "Custom Name",           0,         0, :command_name1],
      :custom2 => [ "Custom Text",          13,         0, :command_name2],
    } # Do not remove this.

    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    # - Shop Data Settings -
    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    # The shop data window displays information about the item in detail.
    # Adjust the settings below to change the way the data window appears.
    #=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    STATUS_FONT_SIZE = 20       # Font size used for data window.
    MAX_ICONS_DRAWN  = 10       # Maximum number of icons drawn for states.

    # The following adjusts the vocabulary used for the data window. Each
    # of the vocabulary settings are self explanatory.
    VOCAB_STATUS ={
      :empty      => "---",          # Text used when nothing is shown.
      :hp_recover => "HP Heal",      # Text used for HP Recovery.
      :mp_recover => "MP Heal",      # Text used for MP Recovery.
      #:tp_recover => "TP Heal",      # Text used for TP Recovery.
      #:tp_gain    => "TP Gain",      # Text used for TP Gain.
      #:applies    => "Applies",      # Text used for applied states and buffs.
      #:removes    => "Removes",      # Text used for removed states and buffs.
    } # Do not remove this.

     WINDOW_POSITIONS = {
      gold_window:      {x: 385, y: 338, width_perc: 40, lines: 1},
      command_window:   {x: 480, y: 0, width_perc: 25, lines: 4},
      buy_window:       {x: 0, y: 123, width_perc: 40, lines: 7},
      sell_window:      {x: 0, y: 123, width_perc: 40, lines: 7},
      status_window:    {x: 0, y: 314, width_perc: 40, lines: 2},
      number_window:    {x: 0, y: 147, width_perc: 45, lines: 6},
      category_window:  {x: 480, y: 120, width_perc: 25, lines: 4},
      help_window:      {x: 0, y: 385, width_perc: 100, lines: 3},
      #data_window:      {x: 0, y: 0, width_perc: 0, lines: 0}
    }

  end # SHOP
end # YEA
```

```
#==============================================================================
=====
# ▼ Editting anything past this point may potentially result in causing
# computer damage, incontinence, explosion of user's head, coma, death, and/or
# halitosis so edit at your own risk.
#==============================================================================
=====

module YEA
  module REGEXP
  module BASEITEM

    IMAGE    = /<(?:IMAGE|image):[ ](.*)>/i

  end # BASEITEM
  end # REGEXP
end # YEA

#==============================================================================
=====
# ■ Numeric
#==============================================================================
=====

class Numeric

  #--------------------------------------------------------------------------
  # new method: group_digits
  #--------------------------------------------------------------------------
  unless $imported["YEA-CoreEngine"]
  def group; return self.to_s; end
  end # $imported["YEA-CoreEngine"]

end # Numeric

#==============================================================================
=====
# ■ Vocab
#==============================================================================
=====

module Vocab

  #--------------------------------------------------------------------------
  # new method: self.item_status
  #--------------------------------------------------------------------------
  def self.item_status(type)
    return YEA1::SHOP::VOCAB_STATUS[type]
  end

end # Vocab

#==============================================================================
=====
# ■ DataManager
#==============================================================================
=====

module DataManager

  #--------------------------------------------------------------------------
  # alias method: load_database
  #--------------------------------------------------------------------------
  class <<self; alias load_database_aso load_database; end
  def self.load_database
    load_database_aso
    load_notetags_aso
  end

  #--------------------------------------------------------------------------
  # new method: load_notetags_aso
  #--------------------------------------------------------------------------
  def self.load_notetags_aso
    groups = [$data_items, $data_weapons, $data_armors]
    for group in groups
      for obj in group
        next if obj.nil?
        obj.load_notetags_aso
      end
    end
  end

end # DataManager

#==============================================================================
=====
# ■ RPG::BaseItem
#==============================================================================
=====

class RPG::BaseItem
```

```
  #--------------------------------------------------------------------------
  # public instance variables
  #--------------------------------------------------------------------------
  attr_accessor :image

  #--------------------------------------------------------------------------
  # common cache: load_notetags_aso
  #--------------------------------------------------------------------------
  def load_notetags_aso
    #---
    self.note.split(/[\r\n]+/).each { |line|
      case line
      #---
      when YEA1::REGEXP::BASEITEM::IMAGE
        @image = $1.to_s
      end
    } # self.note.split
    #---
  end

end # RPG::BaseItem

#==============================================================================
=====
# ■ Game_Temp
#==============================================================================
=====

class Game_Temp

  #--------------------------------------------------------------------------
  # public instance variables
  #--------------------------------------------------------------------------
  attr_accessor :scene_shop_index
  attr_accessor :scene_shop_oy

end # Game_Temp

module SceneManager
  def self.snapshot_for_background
    @background_bitmap.dispose if @background_bitmap
    @background_bitmap = Graphics.snap_to_bitmap
  end
end
```

```
#==============================================================================
=====
# ■ Window_ShopCommand
#==============================================================================
=====

class Window_ShopCommand < Window_HorzCommand

  #--------------------------------------------------------------------------
  # alias method: make_command_list
  #--------------------------------------------------------------------------
  alias window_shopcommand_make_command_list_aso make_command_list
  def make_command_list
    unless SceneManager.scene_is?(Scene_Shop)
      window_shopcommand_make_command_list_aso
      return
    end
    for command in YEA1::SHOP::COMMANDS
      case command
      #--- Default Commands ---
      when :buy
        add_command(Vocab::ShopBuy, :buy)
      when :sell
        add_command(Vocab::ShopSell, :sell, !@purchase_only)
      when :cancel
        add_command(Vocab::ShopCancel, :cancel)
      #--- Imported Commands ---
      when :totorishop
        next unless $imported["KRX-SynthesisShop"]
        process_custom_command(command)
      #--- Custom Commands ---
      else
        process_custom_command(command)
      end
    end
  end

  #--------------------------------------------------------------------------
  # new method: process_custom_command
  #--------------------------------------------------------------------------
  def process_custom_command(command)
    return unless YEA1::SHOP::CUSTOM_SHOP_COMMANDS.include?(command)
    show = YEA1::SHOP::CUSTOM_SHOP_COMMANDS[command][2]
    continue = show <= 0 ? true : $game_switches[show]
    return unless continue
    text = YEA1::SHOP::CUSTOM_SHOP_COMMANDS[command][0]
    switch = YEA1::SHOP::CUSTOM_SHOP_COMMANDS[command][1]
    enabled = switch <= 0 ? true : $game_switches[switch]
    add_command(text, command, enabled)
  end

  #--------------------------------------------------------------------------
  # overwrite method: process_ok
  #--------------------------------------------------------------------------
  def process_ok
    $game_temp.scene_shop_index = index
    $game_temp.scene_shop_oy = self.oy
    super
  end

  #--------------------------------------------------------------------------
  # overwrite method: window_width
  #--------------------------------------------------------------------------
  def window_width; return 160; end

  #--------------------------------------------------------------------------
  # overwrite method: contents_width
  #--------------------------------------------------------------------------
  def contents_width; return width - standard_padding * 2; end

  #--------------------------------------------------------------------------
  # overwrite method: contents_height
  #--------------------------------------------------------------------------
  def contents_height
    ch = height - standard_padding * 2
    return [ch - ch % item_height, row_max * item_height].max
  end

  #--------------------------------------------------------------------------
  # overwrite method: visible_line_number
  #--------------------------------------------------------------------------
  def visible_line_number; return 4; end

  #--------------------------------------------------------------------------
  # overwrite method: col_max
  #--------------------------------------------------------------------------
  def col_max; return 1; end

  #--------------------------------------------------------------------------
  # overwrite method: item_rect
  #--------------------------------------------------------------------------
  def item_rect(index)
```

```
  rect = Rect.new
  rect.width = item_width
  rect.height = item_height
  rect.x = index % col_max * (item_width + spacing)
  rect.y = index / col_max * item_height
  rect
end

#--------------------------------------------------------------------------
# overwrite method: ensure_cursor_visible
#--------------------------------------------------------------------------
def ensure_cursor_visible
  self.top_row = row if row < top_row
  self.bottom_row = row if row > bottom_row
end

#--------------------------------------------------------------------------
# overwrite method: cursor_down
#--------------------------------------------------------------------------
def cursor_down(wrap = false)
  if index < item_max - col_max || (wrap && col_max == 1)
    select((index + col_max) % item_max)
  end
end

#--------------------------------------------------------------------------
# overwrite method: cursor_up
#--------------------------------------------------------------------------
def cursor_up(wrap = false)
  if index >= col_max || (wrap && col_max == 1)
    select((index - col_max + item_max) % item_max)
  end
end

#--------------------------------------------------------------------------
# overwrite method: process_pageup
#--------------------------------------------------------------------------
def process_pageup
  Sound.play_cursor
  Input.update
  deactivate
  call_handler(:pageup)
end

#--------------------------------------------------------------------------
# overwrite method: process_pagedown
```

```
  #--------------------------------------------------------------------------
  def process_pagedown
    Sound.play_cursor
    Input.update
    deactivate
    call_handler(:pagedown)
  end

end # Window_ShopCommand

#==============================================================================
=====
# ■ Window_ShopCategory
#==============================================================================
=====

class Window_ShopCategory < Window_Command

  #--------------------------------------------------------------------------
  # public instance variables
  #--------------------------------------------------------------------------
  attr_reader   :item_window

  #--------------------------------------------------------------------------
  # initialize
  #--------------------------------------------------------------------------
  def initialize
    super(0, 0)
  end

  #--------------------------------------------------------------------------
  # window_width
  #--------------------------------------------------------------------------
  def window_width; return 160; end

  #--------------------------------------------------------------------------
  # visible_line_number
  #--------------------------------------------------------------------------
  def visible_line_number; return 4; end

  #--------------------------------------------------------------------------
  # update
  #--------------------------------------------------------------------------
  def update
    super
    @item_window.category = current_symbol if @item_window
  end

  #--------------------------------------------------------------------------
  # make_command_list
  #--------------------------------------------------------------------------
  def make_command_list
    add_command(Vocab::item,     :item)
    add_command(Vocab::weapon,   :weapon)
    add_command(Vocab::armor,    :armor)
    add_command(Vocab::key_item, :key_item)
  end

  #--------------------------------------------------------------------------
  # item_window=
  #--------------------------------------------------------------------------
  def item_window=(item_window)
    @item_window = item_window
    update
  end

  def draw_item(index)
    rect = item_rect_for_text(index)
    change_color(normal_color, command_enabled?(index))
    draw_text(rect, command_name(index), 1) # The '1' aligns text to center
  end

end# Window_ShopCategory

#==============================================================================
=====
# ■ Window_ShopBuy
#==============================================================================
=====

class Window_ShopBuy < Window_Selectable

  #--------------------------------------------------------------------------
  # overwrite method: item
  #--------------------------------------------------------------------------
  def item
    return index < 0 ? nil : @data[index]
  end

  #--------------------------------------------------------------------------
  # overwrite method: window_width
  #--------------------------------------------------------------------------
  def window_width
    return Graphics.width - (Graphics.width * 2 / 5)
  end

  #--------------------------------------------------------------------------
  # overwrite method: draw_item
  #--------------------------------------------------------------------------
  def draw_item(index)
    item = @data[index]
    return if item.nil?
    rect = item_rect(index)
    draw_item_name(item, rect.x, rect.y, enable?(item), rect.width-24)
    rect.width -= 4
    contents.font.size = YEA1::LIMIT::SHOP_FONT if $imported["YEA-AdjustLimits"]
    draw_text(rect, price(item).group, 2)
    reset_font_settings
  end

end # Window_ShopBuy

#==============================================================================
=====
# ■ Window_ShopSell
#==============================================================================
=====

class Window_ShopSell < Window_ItemList

  #--------------------------------------------------------------------------
  # overwrite method: initialize
  #--------------------------------------------------------------------------
  def initialize(dx, dy, dw, dh)
    dw = Graphics.width - (Graphics.width * 2 / 5)
    super(dx, dy, dw, dh)
  end

  #--------------------------------------------------------------------------
  # overwrite method: col_max
  #--------------------------------------------------------------------------
  def col_max; return 1; end

  #--------------------------------------------------------------------------
  # new method: status_window=
  #--------------------------------------------------------------------------
  def status_window= (window)
    @status_window = window
    call_update_help
  end

  #--------------------------------------------------------------------------
  # new method: update_help
  #--------------------------------------------------------------------------
  def update_help
    super
    @status_window.item = item if @status_window
  end

  def line_height
    24
  end

end # Window_ShopSell

#==============================================================================
=====
# ■ Window_ShopStatus
#==============================================================================
=====

class Window_ShopStatus < Window_Base

  #--------------------------------------------------------------------------
  # alias method: initialize
  #--------------------------------------------------------------------------
  alias window_shopstatus_initialize_aso initialize
  def initialize(dx, dy, dw, dh)
    dh = Graphics.height - SceneManager.scene.command_window.y
    dh -= SceneManager.scene.command_window.height + fitting_height(1)
    dy += fitting_height(1)
    window_shopstatus_initialize_aso(dx, dy, dw, dh)
  end

  #--------------------------------------------------------------------------
  # overwrite method: page_size
  #--------------------------------------------------------------------------
  def page_size
    n = contents.height - line_height
    n /= line_height
    return n
  end

  #--------------------------------------------------------------------------
  # overwrite method: update_page
  #--------------------------------------------------------------------------
  def update_page
    return unless visible
    return if @item.nil?
    return if @item.is_a?(RPG::Item)
    return unless Input.trigger?(:A)
    return unless page_max > 1
    Sound.play_cursor
    @page_index = (@page_index + 1) % page_max
    refresh
  end

  #--------------------------------------------------------------------------
  # overwrite method: draw_equip_info
  #--------------------------------------------------------------------------
  def draw_equip_info(dx, dy)
    dy -= line_height
    status_members.each_with_index do |actor, i|
      draw_actor_equip_info(dx, dy + line_height * i, actor)
    end
  end

  #--------------------------------------------------------------------------
  # overwrite method: draw_actor_equip_info
  #--------------------------------------------------------------------------
  def draw_actor_equip_info(dx, dy, actor)
    enabled = actor.equippable?(@item)
    change_color(normal_color, enabled)
    draw_text(dx, dy, contents.width, line_height, actor.name)
    item1 = current_equipped_item(actor, @item.etype_id)
    draw_actor_param_change(dx, dy, actor, item1) if enabled
  end

end # Window_ShopStatus

#==============================================================================
=====
# ■ Window_ShopNumber
#==============================================================================
=====

class Window_ShopNumber < Window_Selectable

  #--------------------------------------------------------------------------
  # alias method: initialize
```

```
#--------------------------------------------------------------------------
alias window_shopnumber_initialize_aso initialize
def initialize(dx, dy, dh)
  dh = Graphics.height - SceneManager.scene.command_window.y
  dh -= SceneManager.scene.command_window.height
  window_shopnumber_initialize_aso(dx, dy, dh)
end

#--------------------------------------------------------------------------
# overwrite method: window_width
#--------------------------------------------------------------------------
def window_width
  return Graphics.width - (Graphics.width * 2 / 5)
end

#--------------------------------------------------------------------------
# overwrite method: figures
#--------------------------------------------------------------------------
def figures
  maximum = @max.nil? ? 2 : @max.group.size
  return maximum
end

#--------------------------------------------------------------------------
# overwrite method: refresh
#--------------------------------------------------------------------------
def refresh
  contents.clear
  reset_font_settings
  draw_item_name(@item, 0, item_y, true, contents.width - 24)
  draw_number
  draw_total_price
end

#--------------------------------------------------------------------------
# overwrite method: item_y
#--------------------------------------------------------------------------
def item_y
  return contents_height / 2 - line_height * 5 / 2
end

#--------------------------------------------------------------------------
# overwrite method: price_y
#--------------------------------------------------------------------------
def price_y
  return item_y + line_height * 2
```

```
end

#--------------------------------------------------------------------------
# overwrite method: draw_total_price
#--------------------------------------------------------------------------
def draw_total_price
  dw = contents_width - 8
  dy = price_y
  draw_currency_value($game_party.gold, @currency_unit, 4, dy, dw)
  dy += line_height
  draw_horz_line(dy)
  value = @price * @number
  value *= -1 if buy?
  draw_currency_value(value, @currency_unit, 4, dy, dw)
  dy += line_height
  value = $game_party.gold + value
  value = [[value, 0].max, $game_party.max_gold].min
  draw_currency_value(value, @currency_unit, 4, dy, dw)
end

#--------------------------------------------------------------------------
# new method: buy?
#--------------------------------------------------------------------------
def buy?
  return SceneManager.scene.command_window.current_symbol == :buy
end

#--------------------------------------------------------------------------
# new method: sell?
#--------------------------------------------------------------------------
def sell?
  return SceneManager.scene.command_window.current_symbol == :sell
end

#--------------------------------------------------------------------------
# new method: draw_horz_line
#--------------------------------------------------------------------------
def draw_horz_line(dy)
  line_y = dy + line_height - 4
  contents.fill_rect(4, line_y, contents_width-8, 3, Font.default_out_color)
  contents.fill_rect(5, line_y+1, contents_width-10, 1, normal_color)
end

#--------------------------------------------------------------------------
# alias method: update_number
#--------------------------------------------------------------------------
```

```
  alias window_shopnumber_update_number_aso update_number
  def update_number
    window_shopnumber_update_number_aso
    change_number(-@max) if Input.repeat?(:L)
    change_number(@max)  if Input.repeat?(:R)
  end

end # Window_ShopNumber

#==============================================================================
=====
# ■ Window_ShopData
#==============================================================================
=====
=begin
class Window_ShopData < Window_Base

  #--------------------------------------------------------------------------
  # initialize
  #--------------------------------------------------------------------------
  def initialize(dx, dy, item_window)
    super(dx, dy, Graphics.width - dx, fitting_height(4))
    @item_window = item_window
    @item = nil
    refresh
  end

  #--------------------------------------------------------------------------
  # item_window=
  #--------------------------------------------------------------------------
  def item_window= (window)
    @item_window = window
    update_item(@item_window.item)
  end

  #--------------------------------------------------------------------------
  # update
  #--------------------------------------------------------------------------
  def update
    super
    update_item(@item_window.item)
  end

  #--------------------------------------------------------------------------
  # update_item
  #--------------------------------------------------------------------------
```

```
def update_item(item)
  return if @item == item
  @item = item
  refresh
end

#--------------------------------------------------------------------------
# refresh
#--------------------------------------------------------------------------
def refresh
  contents.clear
  reset_font_settings
  return draw_empty if @item.nil?
  contents.font.size = YEA1::SHOP::STATUS_FONT_SIZE
  #draw_item_image
  draw_item_stats
  #draw_item_effects
end

#--------------------------------------------------------------------------
# draw_empty
#--------------------------------------------------------------------------
def draw_empty
  colour = Color.new(0, 0, 0, translucent_alpha/2)
  rect = Rect.new(1, 1, 94, 94)
  contents.fill_rect(rect, colour)
  dx = 96; dy = 0
  dw = (contents.width - 96) / 2
  for i in 0...8
    draw_background_box(dx, dy, dw)
    dx = dx >= 96 + dw ? 96 : 96 + dw
    dy += line_height if dx == 96
  end
end

#--------------------------------------------------------------------------
# draw_background_box
#--------------------------------------------------------------------------
def draw_background_box(dx, dy, dw)
  colour = Color.new(0, 0, 0, translucent_alpha/2)
  rect = Rect.new(dx+1, dy+1, dw-2, line_height-2)
  contents.fill_rect(rect, colour)
end

#--------------------------------------------------------------------------
# draw_item_image
```

```
#--------------------------------------------------------------------------
#def draw_item_image
#  colour = Color.new(0, 0, 0, translucent_alpha/2)
#  rect = Rect.new(1, 1, 94, 94)
#  contents.fill_rect(rect, colour)
#  if @item.image.nil?
#    icon_index = @item.icon_index
#    bitmap = Cache.system("Iconset")
#    rect = Rect.new(icon_index % 16 * 24, icon_index / 16 * 24, 24, 24)
#    target = Rect.new(0, 0, 96, 96)
#    contents.stretch_blt(target, bitmap, rect)
#  else
#    bitmap = Cache.picture(@item.image)
#    contents.blt(0, 0, bitmap, bitmap.rect, 255)
#  end
#end

#--------------------------------------------------------------------------
# draw_item_stats
#--------------------------------------------------------------------------
def draw_item_stats
  return unless @item.is_a?(RPG::Weapon) || @item.is_a?(RPG::Armor)
  dx = 96; dy = 0
  dw = (contents.width - 96) / 2
  for i in 0...8
    draw_equip_param(i, dx, dy, dw)
    dx = dx >= 96 + dw ? 96 : 96 + dw
    dy += line_height if dx == 96
  end
end

#--------------------------------------------------------------------------
# draw_equip_param
#--------------------------------------------------------------------------
def draw_equip_param(param_id, dx, dy, dw)
  draw_background_box(dx, dy, dw)
  change_color(system_color)
  draw_text(dx+4, dy, dw-8, line_height, Vocab::param(param_id))
  if $imported["YEA-EquipDynamicStats"]
    draw_percentage_param(param_id, dx, dy, dw)
  else
    draw_set_param(param_id, dx, dy, dw)
  end
end

#--------------------------------------------------------------------------
```

```
# draw_percentage_param
#--------------------------------------------------------------------------
def draw_percentage_param(param_id, dx, dy, dw)
  if @item.per_params[param_id] != 0 && @item.params[param_id] != 0
    text = draw_set_param(param_id, dx, dy, dw)
    dw -= text_size(text).width
    draw_percent_param(param_id, dx, dy, dw)
  elsif @item.per_params[param_id] != 0 && @item.params[param_id] == 0
    draw_percent_param(param_id, dx, dy, dw)
  else
    draw_set_param(param_id, dx, dy, dw)
  end
end

#--------------------------------------------------------------------------
# draw_set_param
#--------------------------------------------------------------------------
def draw_set_param(param_id, dx, dy, dw)
  value = @item.params[param_id]
  if $imported["YEA-EquipDynamicStats"] && @item.var_params[param_id] > 0
    value += $game_variables[@item.var_params[param_id]] rescue 0
  end
  change_color(param_change_color(value), value != 0)
  text = value.group
  text = "+" + text if value > 0
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
  return text
end

#--------------------------------------------------------------------------
# draw_percent_param
#--------------------------------------------------------------------------
def draw_percent_param(param_id, dx, dy, dw)
  value = @item.per_params[param_id]
  change_color(param_change_color(value))
  text = (@item.per_params[param_id] * 100).to_i.group + "%"
  text = "+" + text if @item.per_params[param_id] > 0
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
  return text
end

#--------------------------------------------------------------------------
# draw_item_effects
#--------------------------------------------------------------------------
def draw_item_effects
  return unless @item.is_a?(RPG::Item)
  dx = 96; dy = 0
  dw = (contents.width - 96) / 2
  draw_hp_recover(dx, dy + line_height * 0, dw)
  draw_mp_recover(dx, dy + line_height * 1, dw)
  draw_tp_recover(dx + dw, dy + line_height * 0, dw)
  draw_tp_gain(dx + dw, dy + line_height * 1, dw)
  dw = contents.width - 96
  draw_applies(dx, dy + line_height * 2, dw)
  draw_removes(dx, dy + line_height * 3, dw)
end

#--------------------------------------------------------------------------
# draw_hp_recover
#--------------------------------------------------------------------------
def draw_hp_recover(dx, dy, dw)
  draw_background_box(dx, dy, dw)
  change_color(system_color)
  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:hp_recover))
  per = 0
  set = 0
  for effect in @item.effects
    next unless effect.code == 11
    per += (effect.value1 * 100).to_i
    set += effect.value2.to_i
  end
  if per != 0 && set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
    draw_text(dx+4, dy, dw-8, line_height, text, 2)
    dw -= text_size(text).width
    change_color(param_change_color(per))
    text = per > 0 ? sprintf("+%s%%", per.group) : sprintf("%s%%", per.group)
    draw_text(dx+4, dy, dw-8, line_height, text, 2)
    return
  elsif per != 0
    change_color(param_change_color(per))
    text = per > 0 ? sprintf("+%s%%", per.group) : sprintf("%s%%", per.group)
  elsif set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
  else
    change_color(normal_color, false)
    text = Vocab::item_status(:empty)
  end
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
end
```

```
#--------------------------------------------------------------------------
# draw_mp_recover
#--------------------------------------------------------------------------
def draw_mp_recover(dx, dy, dw)
  draw_background_box(dx, dy, dw)
  change_color(system_color)
  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:mp_recover))
  per = 0
  set = 0
  for effect in @item.effects
    next unless effect.code == 12
    per += (effect.value1 * 100).to_i
    set += effect.value2.to_i
  end
  if per != 0 && set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
    draw_text(dx+4, dy, dw-8, line_height, text, 2)
    dw -= text_size(text).width
    change_color(param_change_color(per))
    text = per > 0 ? sprintf("+%s%%", per.group) : sprintf("%s%%", per.group)
    draw_text(dx+4, dy, dw-8, line_height, text, 2)
    return
  elsif per != 0
    change_color(param_change_color(per))
    text = per > 0 ? sprintf("+%s%%", per.group) : sprintf("%s%%", per.group)
  elsif set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
  else
    change_color(normal_color, false)
    text = Vocab::item_status(:empty)
  end
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
end

#--------------------------------------------------------------------------
# draw_tp_recover
#--------------------------------------------------------------------------
def draw_tp_recover(dx, dy, dw)
  draw_background_box(dx, dy, dw)
  change_color(system_color)
  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:tp_recover))
  set = 0
  for effect in @item.effects
```

```
    next unless effect.code == 13
    set += effect.value1.to_i
  end
  if set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
  else
    change_color(normal_color, false)
    text = Vocab::item_status(:empty)
  end
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
end

#--------------------------------------------------------------------------
# draw_tp_gain
#--------------------------------------------------------------------------
def draw_tp_gain(dx, dy, dw)
  draw_background_box(dx, dy, dw)
  change_color(system_color)
  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:tp_gain))
  set = @item.tp_gain
  if set != 0
    change_color(param_change_color(set))
    text = set > 0 ? sprintf("+%s", set.group) : set.group
  else
    change_color(normal_color, false)
    text = Vocab::item_status(:empty)
  end
  draw_text(dx+4, dy, dw-8, line_height, text, 2)
end

#--------------------------------------------------------------------------
# draw_applies
#--------------------------------------------------------------------------
#def draw_applies(dx, dy, dw)
#  draw_background_box(dx, dy, dw)
#  change_color(system_color)
#  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:applies))
#  icons = []
#  for effect in @item.effects
#    case effect.code
#    when 21
#      next unless effect.value1 > 0
#      next if $data_states[effect.value1].nil?
#      icons.push($data_states[effect.data_id].icon_index)
#    when 31
#      icons.push($game_actors[1].buff_icon_index(1, effect.data_id))
#    when 32
#      icons.push($game_actors[1].buff_icon_index(-1, effect.data_id))
#    end
#    icons.delete(0)
#    break if icons.size >= YEA1::SHOP::MAX_ICONS_DRAWN
#  end
#  draw_icons(dx, dy, dw, icons)
#end

#--------------------------------------------------------------------------
# draw_removes
#--------------------------------------------------------------------------
#def draw_removes(dx, dy, dw)
#  draw_background_box(dx, dy, dw)
#  change_color(system_color)
#  draw_text(dx+4, dy, dw-8, line_height, Vocab::item_status(:removes))
#  icons = []
#  for effect in @item.effects
#    case effect.code
#    when 22
#      next unless effect.value1 > 0
#      next if $data_states[effect.value1].nil?
#      icons.push($data_states[effect.data_id].icon_index)
#    when 33
#      icons.push($game_actors[1].buff_icon_index(1, effect.data_id))
#    when 34
#      icons.push($game_actors[1].buff_icon_index(-1, effect.data_id))
#    end
#    icons.delete(0)
#    break if icons.size >= YEA1::SHOP::MAX_ICONS_DRAWN
#  end
#  draw_icons(dx, dy, dw, icons)
#end

#--------------------------------------------------------------------------
# draw_icons
#--------------------------------------------------------------------------
#def draw_icons(dx, dy, dw, icons)
#  dx += dw - 4
#  dx -= icons.size * 24
#  for icon_id in icons
#    draw_icon(icon_id, dx, dy)
#    dx += 24
#  end
#  if icons.size == 0
```

```
  #    change_color(normal_color, false)
  #    text = Vocab::item_status(:empty)
  #    draw_text(4, dy, contents.width-8, line_height, text, 2)
  #  end
  #end

end # Window_ShopData
=end
#==============================================================================
=====
# ■ Scene_Shop
#==============================================================================
=====

class Scene_Shop < Scene_MenuBase

  def adjust_window_positions
    windows = {
      gold_window:     @gold_window,
      command_window:  @command_window,
      buy_window:      @buy_window,
      sell_window:     @sell_window,
      status_window:   @status_window,
      number_window:   @number_window,
      category_window: @category_window,
      help_window:     @help_window,
      #data_window:     @data_window
    }

  windows.each do |key, window|
      pos = YEA1::SHOP::WINDOW_POSITIONS[key]
      next unless window && pos

      # Set window width based on percentage
      window.width = (Graphics.width * pos[:width_perc] / 100).to_i
      window.height = window.fitting_height(pos[:lines])

      # Set X position
      if pos[:x].nil? && pos[:width_perc] == 75
        window.x = windows[:command_window].width
      else
        window.x = pos[:x] || 0
      end

      # Set Y position
      if pos[:y]
        window.y = pos[:y]
      else
        # Stack windows vertically based on previous window's position and height
        previous_window = get_previous_window(windows, key)
        window.y = previous_window ? previous_window.y + previous_window.height : 0
      end
    end
  end

  def get_previous_window(windows, current_key)
    keys = YEA1::SHOP::WINDOW_POSITIONS.keys
    index = keys.index(current_key)
    return nil if index.zero?
    prev_key = keys[0...index].reverse.find { |k| windows[k] }
    windows[prev_key]
  end

  alias_method :original_start, :start
  def start
    original_start
    adjust_window_positions
  end

  #--------------------------------------------------------------------------
  # public instance variables
  #--------------------------------------------------------------------------
  attr_accessor :command_window

  #--------------------------------------------------------------------------
  # alias method: start
  #--------------------------------------------------------------------------
  alias scene_shop_start_aso start
  def start
    scene_shop_start_aso
    create_actor_window
#   create_data_window
    clean_up_settings
    relocate_windows
  end

  #--------------------------------------------------------------------------
  # overwrite method: return_scene
  #--------------------------------------------------------------------------
  def return_scene
    $game_temp.scene_shop_index = nil
    $game_temp.scene_shop_oy = nil
    super
  end

  #--------------------------------------------------------------------------
  # alias method: create_gold_window
  #--------------------------------------------------------------------------
  alias scene_shop_create_gold_window_aso create_gold_window
  def create_gold_window
    scene_shop_create_gold_window_aso
    @gold_window.width = Graphics.width * 2 / 5
    @gold_window.create_contents
    @gold_window.refresh
    @gold_window.x = Graphics.width - @gold_window.width
  end

  #--------------------------------------------------------------------------
  # alias method: create_command_window
  #--------------------------------------------------------------------------
  alias scene_shop_create_command_window_aso create_command_window
  def create_command_window
    scene_shop_create_command_window_aso
    return unless SceneManager.scene_is?(Scene_Shop)
    if !$game_temp.scene_shop_index.nil?
      @command_window.select($game_temp.scene_shop_index)
      @command_window.oy = $game_temp.scene_shop_oy
    end
    $game_temp.scene_shop_index = nil
    $game_temp.scene_shop_oy = nil
    @command_window.set_handler(:equip, method(:command_equip))
    process_custom_shop_commands
  end

  #--------------------------------------------------------------------------
  # new method: process_custom_shop_commands
  #--------------------------------------------------------------------------
  def process_custom_shop_commands
    for command in YEA1::SHOP::COMMANDS
      next unless YEA1::SHOP::CUSTOM_SHOP_COMMANDS.include?(command)
      called_method = YEA1::SHOP::CUSTOM_SHOP_COMMANDS[command][3]
      @command_window.set_handler(command, method(called_method))
    end
  end

  #--------------------------------------------------------------------------
  # alias method: create_dummy_window
  #--------------------------------------------------------------------------
```

```
alias scene_shop_create_dummy_window_aso create_dummy_window
def create_dummy_window
  scene_shop_create_dummy_window_aso
  @gold_window.y = @dummy_window.y
  @dummy_window.opacity = 0
end

#--------------------------------------------------------------------------
# overwrite method: create_category_window
#--------------------------------------------------------------------------
def create_category_window
  @category_window = Window_ShopCategory.new
  @category_window.viewport = @viewport
  @category_window.help_window = @help_window
  @category_window.y = @command_window.y
  @category_window.deactivate
  @category_window.x = Graphics.width
  @category_window.set_handler(:ok,     method(:on_category_ok))
  @category_window.set_handler(:cancel, method(:on_category_cancel))
end

#--------------------------------------------------------------------------
# new method: create_actor_window
#--------------------------------------------------------------------------
def create_actor_window
  @actor_window = Window_MenuActor.new
  @actor_window.set_handler(:ok,     method(:on_actor_ok))
  @actor_window.set_handler(:cancel, method(:on_actor_cancel))
end

#--------------------------------------------------------------------------
# new method: create_data_window
#--------------------------------------------------------------------------
#def create_data_window
#  wx = @command_window.width
#  wy = @command_window.y
#  @data_window = Window_ShopData.new(wx, wy, @buy_window)
#  @data_window.viewport = @viewport
#end

#--------------------------------------------------------------------------
# new method: clean_up_settings
#--------------------------------------------------------------------------
def clean_up_settings
  @dummy_window.create_contents
  @buy_window.show
  @buy_window.unselect
  @buy_window.money = money
  @last_buy_index = 0
  @status_window.show
  @sell_window.show
  @sell_window.x = Graphics.width
  @sell_window.status_window = @status_window
end

#--------------------------------------------------------------------------
# new method: relocate_windows
#--------------------------------------------------------------------------
def relocate_windows
  return unless $imported["YEA-AceMenuEngine"]
  case Menu.help_window_location
  when 0 # Top
    @help_window.y = 0
    @command_window.y = @help_window.height
    @buy_window.y = @command_window.y + @command_window.height
  when 1 # Middle
    @command_window.y = 0
    @help_window.y = @command_window.height
    @buy_window.y = @help_window.y + @help_window.height
  else # Bottom
    @command_window.y = 0
    @buy_window.y = @command_window.height
    @help_window.y = @buy_window.y + @buy_window.height
  end
  @category_window.y = @command_window.y
  @data_window.y = @command_window.y
  @gold_window.y = @buy_window.y
  @sell_window.y = @buy_window.y
  @number_window.y = @buy_window.y
  @status_window.y = @gold_window.y + @gold_window.height
end

#--------------------------------------------------------------------------
# new method: show_sub_window
#--------------------------------------------------------------------------
def show_sub_window(window)
  width_remain = Graphics.width - window.width
  window.x = width_remain
  @viewport.rect.x = @viewport.ox = 0
  @viewport.rect.width = width_remain
  window.show.activate
end
```

```
#--------------------------------------------------------------------------
# new method: hide_sub_window
#--------------------------------------------------------------------------
def hide_sub_window(window)
  @viewport.rect.x = @viewport.ox = 0
  @viewport.rect.width = Graphics.width
  window.hide.deactivate
  @command_window.activate
end

#--------------------------------------------------------------------------
# new method: on_actor_ok
#--------------------------------------------------------------------------
def on_actor_ok
  case @command_window.current_symbol
  when :equip
    Sound.play_ok
    $game_party.menu_actor = $game_party.members[@actor_window.index]
    SceneManager.call(Scene_Equip)
  end
end

#--------------------------------------------------------------------------
# new method: on_actor_cancel
#--------------------------------------------------------------------------
def on_actor_cancel
  hide_sub_window(@actor_window)
end

#--------------------------------------------------------------------------
# alias method: activate_sell_window
#--------------------------------------------------------------------------
alias scene_shop_activate_sell_window_aso activate_sell_window
def activate_sell_window
  scene_shop_activate_sell_window_aso
  @status_window.show
end

#--------------------------------------------------------------------------
# alias method: command_buy
#--------------------------------------------------------------------------
alias scene_shop_command_buy_aso command_buy
def command_buy
  scene_shop_command_buy_aso
  @buy_window.select(@last_buy_index)
```

```
#    @data_window.item_window = @buy_window
  end

  #--------------------------------------------------------------------------
  # overwrite method: command_sell
  #--------------------------------------------------------------------------
  def command_sell
    @dummy_window.hide
    @category_window.activate
    @category_window.x = YEA1::SHOP::WINDOW_POSITIONS[:category_window][:x]
    @category_window.y = YEA1::SHOP::WINDOW_POSITIONS[:category_window][:y]
    @command_window.x = YEA1::SHOP::WINDOW_POSITIONS[:command_window][:x]
    @command_window.y = YEA1::SHOP::WINDOW_POSITIONS[:command_window][:y]
    @sell_window.x = YEA1::SHOP::WINDOW_POSITIONS[:sell_window][:x]
    @sell_window.y = YEA1::SHOP::WINDOW_POSITIONS[:sell_window][:y]
    @buy_window.x = Graphics.width
    @sell_window.unselect
    @sell_window.refresh
 #   @data_window.item_window = @sell_window
  end

  #--------------------------------------------------------------------------
  # alias method: on_buy_cancel
  #--------------------------------------------------------------------------
  alias scene_shop_on_buy_cancel_aso on_buy_cancel
  def on_buy_cancel
    @last_buy_index = @buy_window.index
    @buy_window.unselect
    scene_shop_on_buy_cancel_aso
    @buy_window.show
    @status_window.show
  end

  #--------------------------------------------------------------------------
  # alias method: on_sell_ok
  #--------------------------------------------------------------------------
  alias scene_shop_on_sell_ok_aso on_sell_ok
  def on_sell_ok
    scene_shop_on_sell_ok_aso
    @category_window.show
  end

  #--------------------------------------------------------------------------
  # overwrite method: on_category_cancel
  #--------------------------------------------------------------------------
  def on_category_cancel
  @command_window.activate
  @dummy_window.show
  @category_window.x = YEA1::SHOP::WINDOW_POSITIONS[:category_window][:x]
  @category_window.y = YEA1::SHOP::WINDOW_POSITIONS[:category_window][:y]
  @command_window.x = YEA1::SHOP::WINDOW_POSITIONS[:command_window][:x]
  @command_window.y = YEA1::SHOP::WINDOW_POSITIONS[:command_window][:y]
  @sell_window.x = Graphics.width
  @buy_window.money = money
  @buy_window.x = YEA1::SHOP::WINDOW_POSITIONS[:buy_window][:x]
  @buy_window.y = YEA1::SHOP::WINDOW_POSITIONS[:buy_window][:y]
end

#--------------------------------------------------------------------------
# new method: current_command_window_symbol
#--------------------------------------------------------------------------
def current_command_window_symbol
  return @command_window.current_symbol
end

#--------------------------------------------------------------------------
# new method: current_command_window_y
#--------------------------------------------------------------------------
def current_command_window
  return @command_window
end

#--------------------------------------------------------------------------
# new method: command_equip
#--------------------------------------------------------------------------
def command_equip
  show_sub_window(@actor_window)
  @actor_window.select_last
end

#--------------------------------------------------------------------------
# new method: command_synthshop
#--------------------------------------------------------------------------
def command_synthshop
  SceneManager.call(Scene_SynthesisShop)
end

#--------------------------------------------------------------------------
# new method: command_name1
#--------------------------------------------------------------------------
def command_name1
  # Do nothing.
  end

  #--------------------------------------------------------------------------
  # new method: command_name2
  #--------------------------------------------------------------------------
  def command_name2
    # Do nothing.
  end

end # Scene_Shop

#==============================================================================
#
# ▼ End of File
#
#==============================================================================
```

Interview Transcripts:

Interviewee 1:

Q: — 2/28/2025 8:05 PM
Okay~ This is the first script then:
Interview Script 1.txt
First, just give me some initial thoughts, anything that comes to mind
A1: — 2/28/2025 8:08 PM
ok i have no idea what im looking at...
is it to just add special text formatting to the console?
theres a good amount of debug commands though i guess
always helps
what am i doing exactly again
Image
Q: — 2/28/2025 8:09 PM
Right now, just reacting to it but I'll give you the first real question
I'll explain the script after all of the questions
1) What might you add to this script to make it more understandable?
A1: — 2/28/2025 8:10 PM
uh definitely comments for the first part
to keep track of what each script is attempting to do
and how they interact
Q: — 2/28/2025 8:11 PM

2) Is an introductory block always necessary or should a user be able to analyze the code to understand the script?
A1: — 2/28/2025 8:11 PM
yes
ok well maybe not always
but most of them, especially if you are letting others look at it or use it
Q: — 2/28/2025 8:12 PM
3) Should warning blocks be included in scripts?
A1: — 2/28/2025 8:12 PM
warning blocks...?
Q: — 2/28/2025 8:12 PM
As in a block which warns the user not to change something
A1: — 2/28/2025 8:13 PM
oh yes
even just for the creator comments like that are always helpful
Q: — 2/28/2025 8:13 PM
4) Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?
A1: — 2/28/2025 8:14 PM
mmm...depends
left as is and within any bounds specified in comments, there should be debugs or prevention of bugs in the first place
but if someone makes an edit that is like, adding or removing code, they are responsible for that change
stuff like parameter adjustments though should be accounted for
i guess also if you want something highly maleable you should build in debugs...? but im not experienced on that...
im kinda just running my mouth
Q: — 2/28/2025 8:16 PM
5) What parts of the script, if any, might you remove, seeing it as being nonessential?
A1: — 2/28/2025 8:17 PM
um...i havent the slightest clue. debug comments id remove if ive deemed the code complete though
to prevent future console clutter or something
Q: — 2/28/2025 8:17 PM
6) Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?
A1: — 2/28/2025 8:17 PM
they can easily be added back anyway
A1: — 2/28/2025 8:19 PM
uhh...... if some scripts need to rely on others thats ok, but i guess it depends on limitations...? i dont really know anything about this at all....
Image
Q: — 2/28/2025 8:19 PM
I'm more asking whether or not scripts should be allowed to have prerequisites or if they should have all essential code included
A1: — 2/28/2025 8:20 PM
uhh... can you clarify what you mean by prerequisites?

Q: — 2/28/2025 8:22 PM
Several scripts are intended to be used together (often seen in yanfly's, victor's, and many other prominent scripters) and will simply not work if the user does not install the full "system" of scripts, even if they didn't want part of that system.
Many times these systems will have a central module that alters the game engine widely and then more specific customization modules
But using the customization modules requires the central module
A1: — 2/28/2025 8:23 PM
oh....i think it's extremely annoying, can create problems, but sometimes is necessary. very hard to strike a balance there, but if possible, avoid
Q: — 2/28/2025 8:23 PM
Avoid having dependencies you mean? Or avoid gathering code when other modules are available?
A1: — 2/28/2025 8:24 PM
avoid dependencies
Q: — 2/28/2025 8:24 PM
7) Should a script even have a predetermined layout or explanation?
A1: — 2/28/2025 8:24 PM
yes
i feel like that was already asked
with the introduction block
Q: — 2/28/2025 8:24 PM
This one is more asking about whether these formats should be standardized
A1: — 2/28/2025 8:25 PM
yes
this sounds stupid but it also adds brand recognition
like
what i mean is i go through the bs2 scripts and i recognize several scripts from the same author
because of how they format their stuff
i like that guy,.. if only i spoke japanese
Q: — 2/28/2025 8:26 PM
8) Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?
A1: — 2/28/2025 8:26 PM
nah, not necessary
google translate exists now lol
and ai...
Q: — 2/28/2025 8:27 PM
Alright that wraps up the primary questions for this script. Did you have any extra thoughts before I explain what this one does?
A1: — 2/28/2025 8:27 PM
my head hurt
Q: — 2/28/2025 8:27 PM
Lmao
Anything else?
A1: — 2/28/2025 8:28 PM
uhhh

no...
Q: — 2/28/2025 8:29 PM
Alright, this script specifically creates a "console" effect for text in RPG Maker VX Ace, making the displayed text look like it's viewed in a computer terminal or compiler/interpreter. Do you have any extra thoughts now knowing what it does?
A1: — 2/28/2025 8:29 PM
no. that sounds cool
but... wait
isnt that just possible without scripts...?
or am i not thinking correctly
Q: — 2/28/2025 8:30 PM
Not really, this completely changes how the screen looks and reanimates how the text is shown so it looks like it's being typed in real time
You could do this with eventing but it would be simply by displaying images
A1: — 2/28/2025 8:30 PM
i see...
i still bet i can find a way lol
exCESSive use of \> and \< though
Q: — 2/28/2025 8:31 PM
That's the spirit, I'll have to show you it later though
Alright, moving on to the second script:
Interview Script 2.txt
Any initial thoughts?
A1: — 2/28/2025 8:31 PM
oh i recognize this
thats the absolute precision script
Q: — 2/28/2025 8:32 PM
Yep
Any thoughts about its composition though?
A1: — 2/28/2025 8:32 PM
i definitely cant parse it on my own but it can be translated
translating the into will give a clear definition im sure
on how to use
it also seems to have comments throughout the actual scripting
Q: — 2/28/2025 8:33 PM
Assume you can't translate it though
A1: — 2/28/2025 8:33 PM
which could be helpful if you want to edit it
oh
i cant?
Q: — 2/28/2025 8:33 PM
Just for the initial thoughts
A1: — 2/28/2025 8:33 PM
oh
im not
translating

im assuming i can translate it later though and hopefully it'll give me helpful information
but without it im kinda just looking at nothing
i mean i cant understand it
lol
except the self explanatory parts
like definining things in english like "absolute hit"
Q: — 2/28/2025 8:35 PM
What might you add to this script to make it more understandable? (with the same assumption that you can't translate it)
A1: — 2/28/2025 8:35 PM
english.
Image
if i cant do that then nothing ig
Q: — 2/28/2025 8:36 PM
Is an introductory block necessary or should the user be able to analyze the code to understand the script?
A1: — 2/28/2025 8:36 PM
i mean what level of knowledge are we assuming here
Q: — 2/28/2025 8:36 PM
Just in general
A1: — 2/28/2025 8:37 PM
i know some scripts use something like "Word1 = <certain_hit>" which is very helpful when you aint reading allat
i mean this one does
its a bit hard to answer any of these questions cause the script was written in japan. so it should be shown to people who speak the language
or translated
Image
Q: — 2/28/2025 8:38 PM
Should warning blocks be included in scripts?
A1: — 2/28/2025 8:38 PM
yes
my opinions havent changed
Q: — 2/28/2025 8:38 PM
Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?
A1: — 2/28/2025 8:38 PM
uhhhhhhhh.........
user should be responsible for solving errors made by editing the script
if the script is making errors from being imported
thats problem
Q: — 2/28/2025 8:39 PM
What parts of the script might you remove, seeing it as being nonessential?
A1: — 2/28/2025 8:40 PM
from this, nothing
Q: — 2/28/2025 8:40 PM

Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?
A1: — 2/28/2025 8:40 PM
this script hasnt changed my opinions on that question...
Q: — 2/28/2025 8:41 PM
Should a script even have a predetermined layout or explanation? (in terms of standardization)
A1: — 2/28/2025 8:41 PM
yeah
if i get another script like this one
even if japanese i can know what to look for if i know how to use this
Q: — 2/28/2025 8:41 PM
Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?
A1: — 2/28/2025 8:41 PM
user responsible
Q: — 2/28/2025 8:42 PM
Any additional thoughts?
A1: — 2/28/2025 8:42 PM
no
Q: — 2/28/2025 8:42 PM
You already know the script so I don't think I have to re-explain it to you right?
A1: — 2/28/2025 8:42 PM
yeah
Q: — 2/28/2025 8:43 PM
Any thoughts knowing how it's used?
A1: — 2/28/2025 8:43 PM
from a coding persective, no
i have thoughts from a gameplay balance perspective but thats irrelevant
Q: — 2/28/2025 8:44 PM
Okay then, time for script number three:
Interview Script 3.txt
Any initial thoughts?
A1: — 2/28/2025 8:45 PM
english
i know what it does
it's very well detailed
i guess it's on me for not knowing precisely what even some of the comments mean
and some of it's a bit redundant lol
maybe...
Q: — 2/28/2025 8:46 PM
What might you add to this script to make it more understandable?
A1: — 2/28/2025 8:47 PM
Umm, nothing, I suppose
lmfao
Image
Q: — 2/28/2025 8:48 PM

Is an introductory block necessary or should the user be able to analyze the code to understand the script?
A1: — 2/28/2025 8:48 PM
yes
Q: — 2/28/2025 8:49 PM
Should warning blocks be included in scripts?
A1: — 2/28/2025 8:49 PM
yes
and make sure to specify that ignoring these warnings may cause an explosion of the user's head
Q: — 2/28/2025 8:49 PM
Lmao
Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?
A1: — 2/28/2025 8:49 PM
I have no idea if this has a debug or not
same thing i said earlier i suppose
Q: — 2/28/2025 8:50 PM
For reference, as detailed as this one is it has absolutely no debug
A1: — 2/28/2025 8:50 PM
well
if you use at as intended (which it has instructions so you know how its intended)
then it shouldnt create errors and if it does
script writer fail
otherwise no debug
Q: — 2/28/2025 8:51 PM
What parts of the script might you remove, seeing it as being nonessential?
A1: — 2/28/2025 8:51 PM
I noticed the comments are a bit overexcessive
like marking which block the "end" actually ends even when it's barely a few blocks long
that
makes zero sense
uhhh
just
marking every little thing for the sake of marking it just clutters the script
id remove some of that
Q: — 2/28/2025 8:53 PM
Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’? (This one has no prerequisites but you can see from the metadata at the top the writer normalizes having prerequisites for their scripts)
A1: — 2/28/2025 8:54 PM
opinion not changed...
Q: — 2/28/2025 8:54 PM
Should a script even have a predetermined layout or explanation?
A1: — 2/28/2025 8:54 PM
yes still
Q: — 2/28/2025 8:54 PM

Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?
A1: — 2/28/2025 8:54 PM
no still
Q: — 2/28/2025 8:55 PM
Any additional thoughts on this script?
A1: — 2/28/2025 8:55 PM
i think it looks pretty
but does it work
Image
if it does then it's well marked
Q: — 2/28/2025 8:55 PM
It's a bit restrictive but the system does work
A1: — 2/28/2025 8:55 PM
and good self-including introduction/instructions
Q: — 2/28/2025 8:56 PM
As a whole, what level of verbosity would you think is ideal for a script? What features would you expect or not expect in an ideal script?
A1: — 2/28/2025 8:57 PM
I think maybe the installation isnt necessary to have in the actual script, its better put elsewhere
any user interactivity with the script i.e. changing values should be at the top
and ideally it would give you a list of commands, and how to use them
A1: — 2/28/2025 8:58 PM
optional tho
Q: — 2/28/2025 8:58 PM
Any additional thoughts on the research, as a whole?
A1: — 2/28/2025 8:59 PM
uhh
clarity is important
Image
Q: — 2/28/2025 9:00 PM
Can you elaborate on that just a bit?
A1: — 2/28/2025 9:01 PM
knowing how to use a script, and perhaps knowing how it's made, and how to best modify it, is stuff that should be made as apparent as possible even if you arent sharing the script publically
in case you come back to it
or use it as a template idk
Q: — 2/28/2025 9:01 PM
Alright, that's the end of the interview then, thanks for helping me out

Interviewee 2:

Q: — 2/28/2025 9:17 PM

Alright, I'll send you the first script now:
Interview Script 1.txt
Any initial thoughts on it, in general?
A2: — 2/28/2025 9:24 PM
I'm a bit dumbfounded but it's alot of work put into this script for sure

Scene manager code is common appearing in rpgmaker even in 2003

I don't know what "return if" is

And not sure what the difference between update and def update is or dispose is
Q: — 2/28/2025 9:24 PM
I can explain all of those but this is more about the composition and layout, not the code
A2: — 2/28/2025 9:25 PM
Oh
I think it's fine
But you should use slashes //
To categorize in sections for codes

But if it's one whole scriptline, it should be in one category still
Just organise abit more
Q: — 2/28/2025 9:26 PM
1 “What might you add to this script to make it more understandable?”
(again from a user perspective, not from a coding perspective)
A2: — 2/28/2025 9:28 PM
I'd like to put in a bit of side notes beside the different types of codes. To keep track of what I'm looking at

For example I would put a ///changes"example" to this
Q: — 2/28/2025 9:28 PM
2 “Is an introductory block necessary or should the user be able to analyze the code to understand the script?”
Q: — 2/28/2025 9:30 PM
(just for reference, those // are called comments and in ruby the equivalent is #)
A2: — 2/28/2025 9:34 PM
You don't really have to because I have a goldfish memory 😭
If you are a script creator who intends to share his codes, you would usually add side notes beside whatever code to guide them.
Alot of plugins do that
Q: — 2/28/2025 9:34 PM
Yep, notice how this one lacks that though
3 “Should warning blocks be included in scripts?”
A2: — 2/28/2025 9:36 PM
YES
You should have warning blocks, if you accidentally remove a single code, the whole script would be broken

Q: — 2/28/2025 9:36 PM
4 “Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?”
A2: — 2/28/2025 9:38 PM
It kinda depends.
If it's a core engine script, yes it should have a debug system.

But if it's an individual script, the user should fix it themselves. For example, like battle engine, item, etc
A2: — 2/28/2025 9:39 PM
This is so you can get a grasp on what the script does

As a beginner, if you are not knowledgeable on codes, this will help you learn how to edit codes in the long run

But core engines are too big to fix everything which is why a debug system is preferred
Q: — 2/28/2025 9:40 PM
5 “What parts of the script might you remove, if any, seeing it as being nonessential?”
A2: — 2/28/2025 9:42 PM
I don't usually remove codes if I have to because I'm afraid I might mess something up.

Even if it doesn't connect with any other codes, I'll leave it blank for now.

I'd just put // or (# for ruby)
and just put "not used for now"
Q: — 2/28/2025 9:42 PM
6 “Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?”
A2: — 2/28/2025 9:45 PM
I would rather it has modulars because it makes it easier to keep track.

While it is helpful to have scripts that you edit yourself, I think modulars help things cut to the chase and save time

Could be different for others though
Q: — 2/28/2025 9:46 PM
7 “Should a script even have a predetermined layout or explanation?” (In terms of standardization)
A2: — 2/28/2025 9:47 PM
Yeah it should. Especially if it's a script/plugin that changes most things.

But if it's something small, it probably doesn't need it
Q: — 2/28/2025 9:47 PM
8 “Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?”
A2: — 2/28/2025 9:49 PM
Uhhhh.....
Its kinda subjective but again it would be more helpful if you had to share scripts and codes with someone

I for one have japanese scripts and plugins downloaded in my game

I leave it as is and don't translate them
Q: — 2/28/2025 9:50 PM
Okay, did you have any other thoughts on this script?
A2: — 2/28/2025 9:50 PM
You mean the structure or the codes itself?
Q: — 2/28/2025 9:51 PM
Just in general, any thoughts. But that said this research is being shown to people that don't have coding knowledge so sharing certain thoughts about the code probably won't be in the research analysis
A2: — 2/28/2025 9:54 PM
I personally think it could be more categorized and have a bit more side notes

Especially if you want people who don't have coding knowledge to understand better.

But if it's all personal, then you probably don't need to

It's all good really
Abit like how Yanfly does his codes
Q: — 2/28/2025 9:55 PM
For reference I mean the script is being used normally among developers, I just mean the research I'm doing is being shown to people without any coding experience
Alright moving onto the second script:
Interview Script 2.txt
Any initial thoughts?
A2: — 2/28/2025 9:57 PM
Is this how it's supposed to look like??
Image
Q: — 2/28/2025 9:57 PM
No...
A2: — 2/28/2025 9:58 PM
Was it in a different language?
Q: — 2/28/2025 9:58 PM
It's in Japanese
A2: — 2/28/2025 9:58 PM
No wonder
This is definitely a vx ace script
Inside MV, we end scripts/plugins by putting brackets instead like }

If it's actually the end of the entire code you put }:

I remember using VXace and putting "end" at the end of the code

I do notice it's actually categorized and organized in this one which is good
Q: — 2/28/2025 10:02 PM

1 “What might you add to this script to make it more understandable?” (aside from translating it)
A2: — 2/28/2025 10:03 PM
Again, put side notes as a guide line

It's not mandatory but it helps a lot generally
Q: — 2/28/2025 10:03 PM
2 “Is an introductory block necessary or should the user be able to analyze the code to understand the script?”
It's okay if some answers haven't changed
A2: — 2/28/2025 10:05 PM
Same answer
Q: — 2/28/2025 10:05 PM
3 “Should warning blocks be included in scripts?”
A2: — 2/28/2025 10:05 PM
Also same answer
Q: — 2/28/2025 10:06 PM
4 “Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?”
A2: — 2/28/2025 10:06 PM
It seems like a core engine to me even if it's not too big
So yes.
Q: — 2/28/2025 10:06 PM
5 “What parts of the script might you remove, seeing it as being nonessential?”
A2: — 2/28/2025 10:07 PM
I probably wouldn't remove anything
Especially if it's in a different language
Q: — 2/28/2025 10:08 PM
6 “Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?”
A2: — 2/28/2025 10:09 PM
Same thing, it's subjective
Q: — 2/28/2025 10:09 PM
7 “Should a script even have a predetermined layout or explanation?”
A2: — 2/28/2025 10:10 PM
Yes
Q: — 2/28/2025 10:11 PM
Can you elaborate on that or is it the same answer as before?
A2: — 2/28/2025 10:11 PM
It's the same answer
Q: — 2/28/2025 10:11 PM
8 “Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?”
A2: — 2/28/2025 10:11 PM
If the intentions were to share worldwide, yes
Q: — 2/28/2025 10:12 PM
And otherwise?

A2: — 2/28/2025 10:13 PM
If no, then don't need to
It's all up to the creator of the code really on what its really for

If it's just for them, they don't need to translate it just because
Q: — 2/28/2025 10:13 PM
Any additional thoughts on this script?
A2: — 2/28/2025 10:14 PM
It's a nice and well thought out script

Overall organized and easy to understand minus translating
Q: — 2/28/2025 10:14 PM
Alright, here's the last script:
Interview Script 3.txt
Any initial thoughts?
A2: — 2/28/2025 10:17 PM
Yanfly my beloved
This is what I mean by organized and very good briefing

The creator is japanese yet translates English for people to use.
Pretty much everyone who uses rpgmaker uses Yanfly's stuff
The side notes are also there
Q: — 2/28/2025 10:18 PM
1 “What might you add to this script to make it more understandable?”
A2: — 2/28/2025 10:19 PM
I would add just abit more side notes but it's overall nothing really
It's already understandable enough
Q: — 2/28/2025 10:19 PM
2 “Is an introductory block necessary or should the user be able to analyze the code to understand the script?”
A2: — 2/28/2025 10:19 PM
Same answer
Q: — 2/28/2025 10:20 PM
3 “Should warning blocks be included in scripts?”
A2: — 2/28/2025 10:20 PM
Same answer, most of this script already has those too
Q: — 2/28/2025 10:20 PM
4 “Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?”
A2: — 2/28/2025 10:22 PM
The item shop script already has a debug system also so yes
Q: — 2/28/2025 10:22 PM
5 “What parts of the script might you remove, seeing it as being nonessential?”
A2: — 2/28/2025 10:23 PM
No because everything needed is there already and doesn't conflict
Q: — 2/28/2025 10:23 PM

6 “Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?”
A2: — 2/28/2025 10:24 PM
Same answer
This script however doesn't actually need to be modular
Q: — 2/28/2025 10:24 PM
7 “Should a script even have a predetermined layout or explanation?”
A2: — 2/28/2025 10:24 PM
Every script introduction is necessary so yes
Avoiding confusion
Q: — 2/28/2025 10:24 PM
8 “Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?”
A2: — 2/28/2025 10:25 PM
Same answer but for this script, yeah.
Since it was intended to share
Q: — 2/28/2025 10:25 PM
Any other thoughts about this script?
A2: — 2/28/2025 10:26 PM
It's very good and organized
Friendly towards beginners for sure
Q: — 2/28/2025 10:26 PM
Any extra thoughts as a whole towards this research?
A2: — 2/28/2025 10:29 PM
It's pretty interesting research and quite well thought out with how it's structured and its been awhile since I used VXace but I think I'm definitely far more used to java now.
Q: — 2/28/2025 10:30 PM
That concludes the interview then, thanks for your help

Interviewee 3

Q: — 3/2/2025 10:09 PM
Thanks~ so to reiterate what we are doing I'm going to show you a couple scripts and ask about 8 questions about them
The questions are oriented towards the composition elements and layout of the scripts and not the code
Any questions for me?
A3: — 3/2/2025 10:12 PM
Not that i can think of
Q: — 3/2/2025 10:13 PM
Alright then, this is the first script:
Interview Script 1.txt
Do you have any initial thoughts around this script?
A3: — 3/2/2025 10:20 PM
The code is quite neatly organized. Very easy to read.

Q: — 3/2/2025 10:21 PM
1 “What might you add to this script to make it more understandable?”
A3: — 3/2/2025 10:23 PM
Definitely some comments. Someone not as experienced might not understand what some of the functions are doing.
Q: — 3/2/2025 10:24 PM
2 “Is an introductory block necessary or should the user be able to analyze the code to understand the script?”
A3: — 3/2/2025 10:30 PM
I would say it is not necessary for this script.
Q: — 3/2/2025 10:30 PM
Why is that?
A3: — 3/2/2025 10:35 PM
The functions are given clear names, so it's not too difficult to parse the script to understand its basic function. Though not necessary, having an introductory block and/or comments is perfectly acceptable to make it easier to analyze.
Q: — 3/2/2025 10:35 PM
3 “Should warning blocks be included in scripts?”
(as in commented blocks of areas the script writer does not anticipate a user to alter)
A3: — 3/2/2025 10:38 PM
Yes, code the writer deems vital to the script's execution should be made clear as to avoid a user from accidentally altering it.
Q: — 3/2/2025 10:38 PM
4 “Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?”
A3: — 3/2/2025 10:42 PM
I would say it depends on the script. A simple script doesn't particularly need built-in debugging, but more complex scripts should have it, or at least make an attempt at it.
Q: — 3/2/2025 10:42 PM
5 “What parts of the script might you remove, seeing it as being nonessential?”
A3: — 3/2/2025 10:46 PM
The debug outputs, even though they provide information as to functions being successfully called. An advanced user could remove them with little consequence.
Q: — 3/2/2025 10:47 PM
6 “Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?”
A3: — 3/2/2025 10:52 PM
Working out-of-the-box is quite useful for the users that have no intention of altering anything about scripts, and should be the preferred option for the majority of users. Modular scripts as an option should still be encouraged, but are not a requirement.
Q: — 3/2/2025 10:53 PM
7 “Should a script even have a predetermined layout or explanation?” (in terms of standardization)
A3: — 3/2/2025 11:04 PM
Yes, to an extent. Having an easy to follow layout helps beginner script writers create code easier and users to parse the code more efficiently, but you should not enforce standardization as a strict requirement.
Q: — 3/2/2025 11:04 PM

8 “Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?”
A3: — 3/2/2025 11:09 PM
Scripts should be translated as accurately as possible, even if the script writer does not intend for the script to be used by users who would require a translation.
Q: — 3/2/2025 11:10 PM
Any additional thoughts before we move on to the second script?
A3: — 3/2/2025 11:11 PM
No.
Q: — 3/2/2025 11:11 PM
Alright, here is the second script:
Interview Script 2.txt
All of the questions are the same, so I'll start with the script specific questions
Any initial thoughts before we begin?
A3: — 3/2/2025 11:18 PM
The script's introductory block and comments are written almost entirely in a non-English language (in this case, Japanese), making it impossible to read without a translation.
Q: — 3/2/2025 11:18 PM
1 “What might you add to this script to make it more understandable?”
A3: — 3/2/2025 11:22 PM
A translation of the comments and introductory block. There are many comments which presumably explain almost every part of the script, which would make it easier to understand, but are unable to be read by anyone who does not know the language.
Q: — 3/2/2025 11:23 PM
5 “What parts of the script might you remove, seeing it as being nonessential?”
A3: — 3/2/2025 11:28 PM
The introductory block and comments. While they explain how the script functions, and benefit inexperienced users as to how they are to use it, they are strictly nonessential to it.
Q: — 3/2/2025 11:28 PM
As for the other questions, they apply generally to all scripts so you can just say if some of your answers are unchanged
2 “Is an introductory block necessary or should the user be able to analyze the code to understand the script?”

3 “Should warning blocks be included in scripts?”

4 “Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?”

6 “Should scripts be more modular, even though a specific script might have additional dependencies, or should they work ‘out of the box’?”

7 “Should a script even have a predetermined layout or explanation?”

8 “Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?”

A3: — 3/2/2025 11:30 PM
2: Yes, this script is sufficiently complicated enough to require a introductory block to explain the basic use. The rest are unchanged
Q: — 3/2/2025 11:31 PM
Any extra thoughts before we move on to the last script?
A3: — 3/2/2025 11:31 PM
No.
Q: — 3/2/2025 11:31 PM
Interview Script 3.txt
Any initial thoughts on this script?
A3: — 3/2/2025 11:34 PM
The introductory block is quite long and there are many comments near the code blocks the script writer intends to be altered.
Q: — 3/2/2025 11:34 PM
1 "What might you add to this script to make it more understandable?"
A3: — 3/2/2025 11:36 PM
Comments past the warning block, in case advanced users need to edit the main portion of the script for any particular reason.
Q: — 3/2/2025 11:36 PM
5 "What parts of the script might you remove, seeing it as being nonessential?"
A3: — 3/2/2025 11:39 PM
The introductory and warning blocks, and comments. Strictly non-essential, despite explaining how to modify the settings to a user's liking, and denoting where the settings intended to be altered end.
Q: — 3/2/2025 11:39 PM
Same as with script two, the next questions are more general so they may have unchanged answers
2 "Is an introductory block necessary or should the user be able to analyze the code to understand the script?"

3 "Should warning blocks be included in scripts?"

4 "Is it necessary for scripts to have built in debug or should the user be responsible for solving any further errors?"

6 "Should scripts be more modular, even though a specific script might have additional dependencies, or should they work 'out of the box'?"

7 "Should a script even have a predetermined layout or explanation?"

8 "Should scripts be translated to English as a universal language? Or is the user responsible for translating it or determining its use from code?"
A3: — 3/2/2025 11:40 PM
All unchanged from script 2's questions.
Q: — 3/2/2025 11:40 PM
Any additional thoughts on this script or on this study in general?
A3: — 3/2/2025 11:41 PM

The third script goes to quite above and beyond lengths to explain to its user how to operate it properly. This level of user-friendliness should be encouraged more.
Q: — 3/2/2025 11:42 PM
Alright, that should be the end of the interview then, thanks alot for your help~