

## Article

# Worlds of Possibility: Tabletop Roleplaying Open Gaming Licenses Meet Interactive Fiction

Trent Hergenrader

Department of English, College of Liberal Arts, Rochester Institute of Technology, 23 Lomb Memorial Parkway, Rochester, NY 14623, USA; wthgsl@rit.edu

## Abstract

This article argues that students inspired by stories set in enormous open-world videogames can use a combination of free interactive fiction (IF) tools and tabletop role-playing game open gaming licenses (OGLs) to simulate stories that allow their players to customize their characters based on their play style preferences, and to face obstacles that require virtual dice rolls where the player may succeed or fail. This provides them with an experience similar to writing for a major videogame franchise and go through the thought processes required when writing interactive narratives.

**Keywords:** tabletop roleplaying games; interactive fiction; Twine; open gaming licenses

Two of my favorite hobbies, tabletop roleplaying games (TTRPGs) and interactive fiction (IF), have more in common than one might suspect. They've both existed for decades, they rapidly grew in popularity and visibility in the twenty-first century, and yet they remain decidedly outside the mainstream when compared to AAA videogames that incorporate many aspects of TTRPGs and IF. Larian Studios *Baldur's Gate 3*, released in August 2023, has sold over 15 million copies and has been lauded both as a stellar adaptation of *Dungeons & Dragons*, the world's most popular TTRPG, and for being driven by a rich and complex narrative (Mulligan 2024). The ten different companion characters from which players can choose bring depth and variety to the game's narrative, where player decisions have far-reaching consequences for their in-game relationships and for the main storyline. It deservedly won several industry awards for narrative and gameplay and has become one of the first games on people's lips when discussing engrossing narrative-driven games. It is exactly the kind of game my students dream about creating.

As a creative writing instructor in a STEM school with a top-rated game design program for both undergraduate and graduate students, I do my best to keep up with digital gaming trends. Students in my fiction classes are more likely to cite *Baldur's Gate 3*, *Cyberpunk 2077*, *Red Dead Redemption 2*, and games from *The Witcher*, *Elder Scrolls*, and *Fallout* series as their favorite fictional stories, more than even young adult print fiction series like J.K. Rowling's *Harry Potter*, Susanne Collins's *The Hunger Games*, and Stephanie Meyer's *Twilight*. A handful of students may have read game-adjacent novels like Ernest Cline's *Ready Player One*, media tie-ins from the *Star Wars* franchise, or the source material from some of their favorite games, such as the short stories and novels set in *The Witcher* universe created by Polish author Andrzej Sapkowski. But the works of literary writers with speculative elements, like Margaret Atwood, Ann Patchett, Haruki Murakami, or George Saunders? Forget it. It's equally unlikely that they're even familiar with the term



Received: 10 November 2025

Revised: 10 February 2026

Accepted: 25 February 2026

Published: 17 March 2026

**Copyright:** © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

electronic literature, much less familiar with individual authors or canonical works of digital art.

This creates a conundrum for the creative writing professor who wants to simultaneously help these students hone their narrative craft *and* do their part to help elevate the sophistication of videogame narratives. It's akin to teaching a fiction writing workshop where the students want to produce work inspired by their love of Tolstoy's *War and Peace*—a massive work of fiction, whose sheer length makes the complete unwieldy and impractical to use in a typical classroom setting. Coming to grips with a creative project's scope has become a primary challenge for today's student writers, who consume works that exist as part of sprawling transmedia franchises (Jenkins 2006; Murray 1997). Even when they do refer to print fiction, they often refer to "Harry Potter" and mean the entire seven-book series, and not a single novel like *Harry Potter and the Sorcerer's Stone*. In the case of videogames, the problem is compounded; narrative-heavy, open-world exploration games require dozens if not literally *hundreds* of hours to play through the main quest just to get a sense of the game's story arcs and themes, and unique player strategies mean that a game provides many different middles and ends for different types of players.

A traditional novel is read chronologically from front to back with no detours or deviations. The events of chapter 17 are identical for every reader; every reader found their way to chapter 17 by reading the previous 16 chapters sequentially; and the remaining chapters draw from earlier developments in the story. In a novel, the protagonist makes choices that have consequences, and the character arrives at the latter points of the narrative having several experiences and interactions that have influenced their attitudes and beliefs, and thus shape their decision-making leading up to, and through, the climax. In contrast, in a videogame quest, it is the *player* who chooses how to solve problems, traverse obstacles, and otherwise make several non-trivial decisions to reach a midpoint in the main quest (Aarseth 1997). Videogame narratives must be more flexible to accommodate the infinite-seeming choices players and their characters make before arriving at the midpoint in the narrative (Montfort 2003; Salter 2014). Player A focuses only on the main quest missions and arrives at the midpoint after ten hours of play; Player B exhaustively explores every inch of the available map and logs sixty hours of play before reaching the same midpoint as Player A. Neither has played the game more or less correctly than the other, it's just that Player A took a more direct route. The midpoint of the narrative, then, must be adaptable enough to connect to any strategy a player took to arrive there, regardless of whether they chose anti-social, pro-social, or neutral options along the way.

Few narratives in any form of media begin at the protagonist's birth, proceed chronologically, and then conclude with their death; rather, most narrative media start *in media res* in the story of the lives of their characters. Being able to pick up a story at any point is a skill writers need to learn. A fiction writing instructor will encourage the ambitious student writer who is inspired by a massive tome of literature to isolate a single chapter, incident, or passage in the longer work, and then use that smaller bit to examine the different techniques that convey the character's backstory, disposition, and formative life moments. We can use a similar approach with the aspiring videogame writer, picking a specific questline or interaction from a game they find inspiring, and develop this one part with care and precision. Ideally, a well-executed project can later be integrated into a larger work, one possible narrative node among many.

But how can a student videogame writer pick up the narrative *in media res* if their players have arrived at the midpoint having taken different paths to get there? For those previous choices to be meaningful, they need to have consequences for the narrative. The print fiction author avoids this problem because they, as author, make all the decisions on behalf of the protagonist. If I write a short story that will eventually serve as chapter

10 of 20 in a future novel, I would need to think through what mental state and stage of development my protagonist will have reached at that midpoint in the narrative. I, as the fiction writer, decide whether the protagonist will have taken the moral high ground or perhaps compromised their morals; I can use literary techniques such as allusions, internal thoughts, conflicting emotions, and dialog to convey to the reader where the protagonist currently stands and provide context that suggests how they got there. What can we do if it's the *player* who has made those decisions on behalf of their character, and that different players will have made different choices? While, in theory, it might be good practice for the student to write the scene multiple times, each one with the protagonist in a different headspace or with contrasting attitudes toward the world, this would be inefficient and likely an uninspiring task for the student writer, not to mention the time constraints of working within a semester.

To solve this, I take an approach that retains key aspects of player choice and consequences reminiscent of large-scale videogames while simplifying the development process by combining aspects of IF and TTRPGs. The integration of TTRPG Open Gaming Licenses (OGLs) with Interactive Fiction (IF) authoring tools provides more than just a technical shortcut; it offers a specific rhetorical framework for teaching dynamic storytelling. By leveraging established game engines, student writers can move beyond basic branching paths to create “literary” depth—where a story reacts not just to player clicks, but to a protagonist's established identity and the consequences of their previous failures or successes. A single author can provide their IF players with a series of options that allow them to customize their protagonist's attitude, personality, and character customization options at the start of the narrative and allow these choices to have a meaningful impact on possible pathways forward (Reed et al. 2020). This process, which I argue maintains the complexity of rich, well-written fiction and diverse choices in customizing the protagonist and narrative without needing an army of developers and a nine-figure budget (Anthropy 2012; Freese et al. 2024; Reed 2023), can be accomplished through a combination of employing open-gaming licenses (OGLs) from popular TTRPGs and free, easy-to-learn interactive authoring tools such as Twine or Ren'Py, or Inform or Unity for writers with more advanced programming skills.

## 1. The Rise of Open Gaming Licenses

TTRPG designers create a system of interconnected rules that determine how challenges are resolved in their games. If a hero wants to battle an ogre, haggle with merchants, investigate for clues, or any one of an infinite number of actions, the game's referee, usually called the dungeon master (DM) or game master (GM) must rule what happens next. Mundane actions—climbing a normal staircase, driving across town, eating a chicken sandwich—automatically succeed, and the GM will often allow trivial difficulties to pass without any additional action needed. For example, a hero would have no problem restraining a young child or a master thief could slip past a sleeping guard with no fear of being detected. For more challenging or contested actions—climbing a collapsing staircase during a fire, speeding across downtown at 90 miles per hour, or eating a chicken sandwich while the chicken is still alive—could fail as easily as they could succeed. This is when the GM invokes the game's rules and directs the players on what result they need to succeed. Though a minority of games use cards, coins, or the game rock-paper-scissors to resolve the action, the majority of TTRPGs use dice to determine an action's success or failure.

The number and type of dice vary widely between games. A common TTRPG dice set contains six polyhedral dice, indicated in writing by the lowercase letter d followed by the number of sides of the die. A common six-sided die thus is referred to as d6. A typical TTRPG dice set includes a d4, d6, d8, two d10s, d12, and d20; a GM might tell a

player to roll 3d6, meaning three six-sided dice, or more complicated combinations, such as rolling 2d4, 2d6 and a single d10. The dice grouped together in something called a dice pool, and they are rolled simultaneously. The GM uses results from a rolled die or dice pool to determine the result of the action. The game rules dictate the quantity and type of dice rolled, depending on a wide variety of factors, and in most cases require players to reference their character sheet, where their characters have different skills and abilities. In a classic D&D game, a hero shooting a crossbow at a goblin would first check their character sheet for their dexterity ability and other modifiers for bonuses or penalties—special skills, enhanced equipment, or poor visibility or an enemy hiding behind cover—and then they roll a d20 and add the modifiers to the number rolled. The player must roll above a value that represents their opponent's defensive capacities, called armor class (AC), a value that is often known only to the GM.

In the above scenario, the player's character has a score of 16 of a possible 18 in their dexterity, which grants them a +3 to their roll. The character is proficient with a bow, which adds another +2, for a total of +5. The player rolls a d20 and gets a 14, adds the +5 bonus, and announces that the final result is a 19. The GM consults the rulebook, sees that the goblin has an AC of 15. The player's original dice roll, 14, was below the goblin's AC and thus would have missed, but their +5 bonus brings the total up to a 19; the GM tells the player that she hit her target and may now roll to see how much damage the bolt causes. The player rolls the appropriate damage dice for the weapon, which in this case is 1d8 plus the character's dexterity modifier. The player rolls the d8 and gets a 6, then adds the +3 from their dexterity modifier, and does 9 points of damage. The GM again references the opponent's stats, this time their hit points (HP) that determine the creature's overall health. This goblin has 7 HP, so a crossbow bolt that deals 9 damage means the goblin's health has been reduced to -2 and is either incapacitated or dead.

Players familiar with TTRPGs likely skimmed past the above paragraph, while someone who has never played a TTRPG might need to read the paragraph twice or more to follow the math. Keep in mind the above is a *simple* example, and as characters gain experience and improve their skills and equipment, calculating all the modifiers can be a time-consuming challenge. With no exaggeration, we could add bonuses for a magic crossbow bolt, shooting from a flanking position, shooting from an elevated position, etc., and subtract penalties for the opponent's shielding spell, generally poor visibility due to rain, and for the goblin being partially concealed behind a tree. While some players enjoy the tactical side of such combat, others quickly grow bored at the long delays as the GM searches through four different manuals to calculate all the modifiers for the challenge.

Games that prioritize storytelling over combat simplify this by reducing the number of possible modifiers and using fewer types of dice; for example, some systems only use d6s, or d10s, or d20s. Different rule sets instruct players to read the dice differently too. Sometimes all the rolled dice will be added together, a few dice will be multiplied, or each die is counted separately, or only a certain value count. For example, rolling 3d6 with the results of 1-3-6 could either total 10 ( $1 + 3 + 6$ ) or 18 ( $1 \times 3 \times 6$ ), or the action was a success because at least one die landed with the 6-side face up, or it could be a failed result because rolling a 1 neutralizes the 6. The point of using fewer dice and limited modifiers allows the GM and player to resolve the action roll quickly and jump straight back into more freeform narrative storytelling.

Instructing players on how to resolve conflicts is a critical part of a game's core rule set, which are also referred to as its *game mechanics* or the *game engine*. As a matter of legality, game mechanics cannot be copyrighted (Schaeffer n.d.); however, a TTRPG game's setting and unique proper nouns such as the names of specific spells or weapons can be, though this can lead to some murky situations where the law is not clear. For example,

“magic missile” is a popular lower-level spell in D&D, yet the idea of a wizard shooting burning rays at an enemy is not original to the game. Thus, a designer of a different fantasy game could name a spell “magic missile” and avoid being sued over it, especially if its description varies significantly from that in the D&D rules. In practice, Wizards of the Coast (D&D’s publisher) is more concerned about protecting the copyright of their unique campaign settings, such as *Forgotten Realms*, *Eberron*, and *Ravenloft* and the characters, locations, and histories of these fictional places.

In fact, Wizards of the Coast was the first TTRPG company to allow and even encourage players to use their rule system to create their own content under an open gaming license or OGL ([Open Gaming Foundation n.d.](#)). TSR Hobbies, the original company to publish D&D, went bankrupt in 1997. Wizards of the Coast bought D&D and, in 2000, reversed TSR’s history of cracking down on fan creators of D&D content. They instead published a Systems Reference Document (SRD) that outlined what players could freely use to develop their own game products for D&D. The logic was that player-created content would serve to fuel even higher sales of the licensed D&D content published by Wizards of the Coast. The OGL identified what creators could use without needing permission and what specific game elements retained their copyright, and the SRD was a guide to the game’s core mechanics and conflict resolution system.

Over the last 24 years, more and more game publishers have begun adopting OGLs and sharing SRDs with an aim of getting more fan-made content on the market to boost sales of the licensed material. Today, in 2025, dozens of OGLs exist with rulesets that can be freely adopted for gamers who use them to create their own unique settings. Companies publishing their OGLs and SRDs on their websites for players to access freely, and the rise in the number of sites where creators can sell their content has democratized TTRPG design in significant ways. Why create a whole new rule system for a new TTRPG and worry about balance and clarity of the instructions when instead one can simply use the engine of a game they admire? Furthermore, a handful of popular OGLs and SRDs have spawned hundreds of games using those systems. Some of the more popular OGL/SRDs include systems like *Powered by the Apocalypse*, *Forged in the Dark*, the *Year Zero Engine*, Monte Cooke’s *Cypher System*, *Fate Core*, *OpenD6*, and *Gumshoe*, all of which have their own feel and can be better suited to certain types of games. For example, *Powered by the Apocalypse*, the *Cypher System*, and *Fate Core* offer a great deal of flexibility for all types of games, where *Gumshoe* is meant for detective-themed games and solving mysteries; the *Year Zero Engine* incorporates fewer but more impactful rolls, and *Forged in the Dark* is good for “failing forward” games, where players often succeed in their actions but even those successes include a catch or contain negative consequences.

Designers are encouraged to tweak the rules to their tastes too. There are literally hundreds of *Powered by the Apocalypse* games that span every genre of fiction imaginable but no two are identical; both *Monster of the Week* (inspired by television shows like *Buffy the Vampire Slayer* and *Supernatural*) and *Cartel* (inspired by shows like *Narcos* and *Breaking Bad*) use the *Powered by the Apocalypse* system, but the former has details for magic powers and spectral horrors while the latter includes rules for cooking drugs and bribing police. Another benefit of having a group of popular OGL/SRDs is that once a player learns the core rules, they can quickly pick up any game that uses the same engine. For example, *Free League’s Year Zero Engine* uses only d6s that players roll in pools, and a player must get a 6 to succeed ([Härenstam n.d.](#)). Though it can be counterintuitive for novice players, a 5d6 roll of 5-5-5-5-5 would fail where a 1-1-2-1-6 would be successful. The main modifier in the *Year Zero Engine* is the adding or subtracting of d6s from the dice pool rather than dealing with different values, so a player who would normally roll 5d6 can use a weapon for +1d6 for a total of 6d6, or they’re injured and suffer a −1d6 penalty, meaning they’d roll

a 4d6. Once a player understands this core mechanic, they can quickly pick up and play any Year Zero Engine game, from Free League's *Alien* (based on the horror franchise of the same name), or *Tales from the Loop* (a cross between the film *Goonies* and the Netflix series *Stranger Things*) or *Vaesen* (Swedish 19th-century folk horror). The basic rules of the game engine remain the same regardless of the game's setting or genre.

In short, TTRPG designers have a wealth of options when it comes to choosing a game engine rather than being forced into designing their own. Now they can select the OGL/SRD elements that best suit their game best and then tweak and modify the core rules through additions, subtractions, and combinations of different rule sets. Therefore, creators can begin designing their games with a proven and well-balanced game engine that will already have a built-in audience of games that use a similar rule set. For example, game designers who use the Powered by the Apocalypse system will often market their games on crowdfunding sites and platforms like Discord to a sizeable audience of players who already purchased, played, and enjoyed other Powered by the Apocalypse games.

## 2. Interactive Fiction Authoring Tools

As with TTRPG game engines, writers of digital literature and IF have a range of free authoring tools, with software like Twine, Ren'Py, Unity, and Unreal as open-source options, or they can use free but proprietary tools like Inklewriter and ChoiceScript. As with the tabletop rule sets, these tools vary in the number of different features they offer, and the level of coding knowledge required to create a game. All of them allow for authoring choice-based narratives where the player makes decisions on the actions the character will take; they can also be enhanced with images, audio, and other media, and they all can store variables. This allows players to have an interactive inventory from which they can claim and drop objects, monitor different stats such as their character's health or the status of their relationships with other characters, or a host of other functions.

Twine is one of the most flexible free solutions that provides options for the absolute beginner all the way up to the sophisticated programmer (Salter and Moulthrop 2021). The basic features of Twine can be taught to a child in a few minutes, showing them how to create linked and branching passages for a simple story. It uses a simple graphical interface where the user creates blocks called passages that contain a blend of programming commands, which are hidden from the player when the game is played, and the text that will be visible to the player. Passages can be arranged on the screen in Twine in any configuration, and each card can hold a large amount of text. One limitation is that each passage must have a unique name as its title. For this example, each passage represents a room in the mansion, so one passage is titled "foyer," another "library" and the last one "dining room." In Twine, I would set the default passage as "foyer," meaning every game session starts with the foyer passage. In the Twine Sugarcube (v2.37.3, released on 26 July 2024) version of the software (other versions exist that have different commands and functionality), authors create links between passages by placing double brackets around words or groups of words. The brackets indicate to which passage the link leads. My text in "foyer" might read like this:

You stand in the foyer of the haunted mansion. The door behind you to the outside has been bolted fast, giving you no option but heading deeper inside the building. From here, you can either head to the [[library]] or the [[dining room]].

On the player's screen when playing the game, the words "library" and "dining room" would be clickable links that would forward the player to one of those two passages. New passages can also be conveniently created inside existing passages as well. In this scenario, if I decided while writing the passage in the Twine editor that I wanted to add a third option, a game room, I could simply amend the last sentence to include "[[library]], [[dining

room]], or [[game room]]” and Twine would automatically create and link to a new blank passage named “game room.”

Twine makes storing variables relatively simple for beginners as well. If in the above passage, the player advances to the library, they would see the following passage on their screen.

Cobwebs cluster in the corners of the dilapidated library. A thick coat of dust lines the bookshelves. An old writing desk stands in the center of the room, also covered in dust. Laying on the desk, you see a diary and a butcher’s cleaver.

In this example, the diary and butcher’s cleaver appear as hyperlinks. Clicking on them allows for a unique interaction where they could add either item, or perhaps both, to their character’s inventory using an if–then statement:

```

    <<if !$chosenItem>>
    <<link “Take the diary”>>
        <<run $chosenItem = “diary”>>
        <<take “diary”>>
        <<goto “Library”>>
    <</link>>
    <br>
        <<link “Take the butcher’s cleaver”>>
        <<run $chosenItem = “butcher’s cleaver”>>
        <<take “butcher’s cleaver”>>
        <<goto “Library”>>
        <</link>>
<<else>>
    <<if $chosenItem == “envelope”>>
        The envelope is now tucked in your pocket.
    <<elseif $chosenItem == “butcher’s cleaver”>>
        You heft the cleaver. Dried blood flakes off the blade.
    <</if>>

```

The player then advances further in the story, armed with the diary and the cleaver in their inventory. Games with these options contain links for players to open their current inventory and inspect the contents. In this example, the diary could be read for clues for solving puzzles, and the cleaver can be used to break open containers after Twine verifies you have the necessary item in your inventory:

```

    <<if $inv.includes(“butcher’s cleaver”)>>
        You can smash the crate open with your cleaver.
    <<else>>
        The crate is made of heavy wood. You need something to help open it.
    <</if>>

```

Even a simple inventory system can add many options and choices for the writer designing the story and thus the end player. For example, the player may need to hunt through the mansion until they collect a list of objects, and they will only be able to unlock the next chapter of the story once they possess them all in their inventory.

Tracking a player’s health, or hit points, can also be easily handled in Twine without much programming knowledge. At the start of the game, this single line of code sets the variable for maximum hit points at 12: <<set \$hpMax = 12>>. If the player’s choices lead their character to fall down a flight of stairs, another line of code modifies the variable: <<set \$hp -= 2>>. This deducts two points of damage, bringing the character’s hit points to 10. This status could be made visible to characters all the time as a number in a status bar, or instead show a narrative description without the player being aware of the precise

value, e.g., “Your body aches and you’ll be nursing some bruises in the morning, but you’re still standing.” As the player’s hit points drop, the messages can become more dire, e.g., “You’re bleeding and wheezing badly. You can’t take any more of this!” and if the character reaches zero hit points, the game instantly jumps to a passage that says the character has expired and the player can either quit or start a new game. This straightforward system can also interact with items in the inventory, where a first aid kit could be collected and later deployed to boost the character’s hit points by 2, e.g., `<<set $hp += 2>>`. Allowing the character’s health to fluctuate up and down adds tension to a story, especially if a streak of poor choices left their character in a weakened position as they head toward the story’s conclusion.

Using Twine to track hit points and weapons implies that the story’s action will involve conflict and violence, but this isn’t the only use for storing variables; instead of hit points, the creator could create different variables for whatever type of game they want to design. This could be emotions (meters for tracking to what degree the character is happy, frightened, sad) or different types of quality of life (physical health, mental health, social relationships) with lines of code that add or subtract from numeric values, and multiple possible end states, e.g., the game jumps to an end passage once a player’s happiness score is greater than 45 or if their mental health drops below 10. The code that adjusts the different tracks can be triggered by certain actions, e.g., choosing the option to work out boosts physical health; choosing to go out drinking with friends boosts social relationships but lowers physical and mental health. No matter how many variables you choose, the same basic code blocks can be used repeatedly with minor tweaks.

These three IF strategies—creating links between passages, designing a simple inventory system, and tracking the changes for certain variables—can be learned by anyone who has a basic understanding of computing. Dozens of forums for questions and answers exist for all these IF tools, so if someone new to programming cannot discover why the code is not working in the way they intended, more experienced programmers can spot problems or suggest alternatives for achieving a similar outcome. Twine’s features also scale with a writer’s programming and media knowledge. Twine stories can become quite complex, incorporating JavaScript, images and audio, and many other built-in macros and features.

### 3. Tabletop Roleplaying Game Rules Meet Interactive Fiction

As previously mentioned, many learners struggle with finding an appropriate scope for their creative project. This is true in all forms of media. A student charged with writing a 4000-word short story will often attempt to squeeze in a novel’s worth of characters and plotlines; a student assigned with making a five-minute film dreams up a concept that would take 30 min to clearly convey. Of course, the beginner doesn’t know what they don’t know, and most often they are basing their stories on narrative media that they’ve consumed. If students enjoy reading print fiction, they will almost always read novels—quite often ones that belong in a multibook series—and not short stories, so the rhythms and space of that brief format are unfamiliar to them. Likewise, unless a student has watched dozens of five-minute films, they will likely use television episodes or feature films as their models for visual storytelling. Especially in introductory classes, writers learn the hard way that trying to stuff too much story into too small a vessel results in a confusing mess rather than a profound narrative. They often need to attempt to realize their narrative ambitions and fail before they finally see that it’s better to execute a small, concise project that does a few things well rather than having a sprawling but confusing and emotionally shallow experience.

Instructors also have the problem of students spending more time honing their technical expertise than developing their storytelling skills. This is never truer than when

teaching a tool like Twine to computer science majors. Their instinct is often to understand how to squeeze the most out of the tool rather than thinking about the fundamentals of fiction. Left to their own devices, many students would spend their time developing a sophisticated combat engine that handles dozens of modifiers for the player's character rather than asking more fundamental questions of what makes for a compelling character or an interesting narrative choice.

With these free tools, writers can combine the best of OGLs from TTRPGs with the basic features of an IF tool like Twine to create engaging and complex fiction without needing to be experts in game design or programming or spending money on special software. I have been experimenting in my classrooms with building dice-rolling engines based on the rulesets from OGLs, with the concept that any OGL can be saved as text file and imported into an IF tool along with instructions on how to embed dice rolls into the narrative.

I worked on a project where two students adapted the version of the Year Zero Engine used for the mystery game *Vaesen* in Twine as a proof-of-concept prototype, but the premise would work for any of the OGLs, especially ones that use a simple system for basic challenge resolutions and use a single type of die. IF tools are entirely capable of handling multiple die types and dozens of different modifiers, but OGLs tend to be more narrative-, rather than combat-, focused. The Year Zero engine has different versions for different games, but they all follow a similar principle in that players add two values—an attribute value and a skill value—to build a d6 dice pool. *Vaesen* features 4 primary attributes (Physique, Precision, Logic, and Empathy) that range between 2 and 5, and 12 skills whose values range from 0 to 3 (Hintze 2020, pp. 18–21). Players distribute 14 points between their 4 attributes and another 12 points between the 12 skills. To resolve a challenge in the game, the player describes the action they want to make and then combines the attribute and skill for that action for their roll. They add those two values together and roll that quantity of d6s, hoping to see at least one die land with the 6 showing, indicating their success.

In Twine, the initialization that sets a character's attributes and skills looks like this:

```
(set: $attributes to (datamap: "Physique", 2, "Precision", 5, "Logic", 3, "Empathy", 3))
```

```
(set: $skills to (datamap: "Agility", 3, "Close Combat", 2, "Force", 2, "Medicine", 6, "Ranged Combat", 5, "Stealth", 8, "Investigation", 5, "Learning", 5, "Vigilance", 5, "Inspiration", 4, "Manipulation", 3, "Observation", 5))
```

The datamap values change based on which character the player selects. On the opening screen of the game, the player must choose a character based on one of three links: [[Vagabond]], [[Bruiser]], [[Scholar]]. The link leads to passages with those names and sets the appropriate variables. A player can compare the different characters, resetting the variables in the datamap each time until they click the option of [[Begin the Game]] at the bottom of each character sheet.

While this might sound complex, in practice it is quite simple. Jackie's character is a vagabond who wants to sneak into a warehouse guarded by two nightwatchmen. The rules state that such a clandestine action requires a successful Stealth roll, and in *Vaesen* this number is derived from combining the character's Precision attribute and their Stealth skill. Jackie has a 4 in Precision and a 2 in Stealth, meaning she rolls a dice pool of 6d6. Her roll results in one die being a 6, meaning the stealth action is successful and she has slipped past the guards unnoticed. Unlike D&D, *Vaesen* usually only employs a few modifiers for any given roll, and the modifier usually adds or subtracts a single d6 to the dice pool. In a slightly more complex scenario, the GM may decide that the foggy night gives Jackie's character an advantage (+1d6 for 7d6) or that an earlier failed roll puts the guards on high alert (−1d6 for 5d6). Assembling the dice pool takes mere seconds and determining if the

roll succeeded or failed takes an instant, allowing the players to avoid number crunching and focus on descriptive storytelling.

Since there are only 12 possible skills in *Vaesen*, players must fit whatever action they wish to take into those twelve options. While some skills are cut and dried, e.g., Ranged Combat is always used for thrown or projectile weapons, others are more open to interpretation. If Jackie's character is trying to bluff the police chief into letting her view criminal files, is it an Inspiration or Manipulation skill roll? In *Vaesen*, it would depend on the character's disposition. If they were more likely to flatter the chief, it would be an Inspiration roll, whereas if they were apt to lie to him, it would be Manipulation. Players tend to choose their strategies based on what will let them roll the most dice and hence have a greater chance of success, and thus it encourages them to play in a manner consistent with the way they created their character. A player with a 3d6 dice pool for Inspiration and a 5d6 dice pool for Manipulation would usually choose the latter for the better odds mechanically, but in-game, this means their persuasive character would more often act like a charlatan than a flatterer. With these rules deliberately written to be open to interpretation, players must negotiate what every given roll means in the context of their characters and their specific game.

The pared down *Vaesen* version of the Year Zero Engine developed by my students for their Twine prototype eliminates some of the game's more advanced mechanics—like talents that can only be used for bonuses in a very specific situation, or allowing each player to pick a one-time advantage for a mystery—in order to simplify the programming knowledge required for anyone who wishes to adopt their game engine. The dice roller is a short text file titled "Roller" that can be uploaded or copied and pasted into Twine that automates the basic mechanics.

The code sets a temporary value for the dice roll to zero at the start to delete any previous rolls and draws from the appropriate value from the character's sheet as \$skilltest, which will be an integer that is the sum of the character's attribute and skill for that skill test, usually a number between 2 and 8. The roller then rolls the appropriate number of d6s and returns a random value from 1 to 6. Results 1 to 5 do nothing, but a roll of a 6 sets the \$temp variable to +1. This means a player can receive multiple successes on the same roll, where \$temp could be 1, 2, 3 or potentially more. In this example, instead of simply showing the number, it instead displays an image of a die face, d1 to d6, to better simulate an actual roll. If \$temp is at least 1, then the \$outcome variable is set to "Success." If \$temp is greater than 1, the \$outcome is set to "Critical Success" indicating an even better result, and if \$temp is zero, the \$outcome is set to "Failure."

In the passage, the writer only needs to paste the text below to create an option for rolling the dice. These five lines of code embedded provides choices for the player to pick the appropriate skill, and then sets that variable as \$skilltest based on the values on the character sheet. Next, it carries that variable to the Roller passage and switches the \$diceroll variable to the correct number of dice, generates a number of random values, checks to see if any random numbers equal a 6, and then returns the \$outcome as "Success," "Critical," or "Failure."

For each outcome, the IF writer replaces the simple "print: result" with more vivid text. In this case, instead of simply displaying "Success," it would print, "Holding your breath, you crouch and creep under the window, avoiding the shaft of light it casts into the night. You steal a glance back at the watchman, who continues to smoke his cigarette, oblivious to your presence," where as a failure would show, "You take a single tentative step beneath the window and hear an audible snap of a stick under your foot, and you wince. The guard bolts from his chair and shouts, *Who goes there?*"

The *Vaesen* rulebook provides a table that shows the chance for success given the number of d6s rolled (Hintze 2020, p. 41). Rolling 4d6 has a 52% chance—essentially a coin toss—of getting at least one six; rolling 6d6 raises the chances to 67%, while 2d6 has only a 31% chance. Players are thus encouraged to make choices that will give them the best chance of success and avoid options with longer odds, encouraging them to play in character for a higher chance of success. Using the above example, Jackie’s persuasive character will be inclined to solve problems with their silver tongue, while Jordan’s physical character is more likely to use their fists.

The IF writer doesn’t need to understand exactly how the code works; they simply need to copy/paste the code and follow the instructions of where to edit the code in their passages. For example, a beginning IF writer would first need to upload the Roller text as a separate passage with that exact name. This passage sits by itself, not directly linked to any other passages, and is only referenced when needed to resolve a challenge. Second, the IF writer would need to paste the datamap into one or more character sheets that represent different archetypes and have different values for their attributes and skills, which the player chooses before beginning the game. Then, throughout the game, the IF writer can present different options for attempting to overcome a challenge by pasting the text above that sets the \$skilltest to the appropriate variable in the datamap and then calls the Roller passage. When listing the different options, the IF writer must be careful to follow the exact syntax of the variables as listed in the datamap, e.g., “Stealth” and not “stealth” or “sneak”, or the code won’t function correctly. Other than that, once the Roller passage exists and the player selects a character sheet to set the skill variables, the writer can choose as many or as few options they want for each obstacle. All they need to do is paste in the above code and edit the \$skilltest according to the skill used for each option.

As a matter of practicality, the IF writer must narrow the choices down to the few most likely options a player may wish to take but, using the Year Zero Engine, the formula is always the same: ability score + skill score = d6 dice pool. Let’s return to the example of Jackie’s vagabond and her character’s desire to gain access to the restricted warehouse. We already saw how the ability and skill combine for sneaking (Precision 4 + Stealth 2 = 6d6), but other options exist. She could try to bluff her way past the guards (Empathy 2 + Manipulation 2 = 4d6) or pick them off with a blowgun (Precision 4 + Ranged Combat 1 = 5d6), or charge in fists flying (Physique 2 + Close Combat 0 = 2d6). The IF writer can offer all these options and more, provided that the software can retrieve a character’s abilities and skills, add them together, and roll a d6 dice pool based on the two values combined.

On the back end of the IF, this is accomplished by the initialization file that sets the variables for the character archetype at the start of the game. In this way, when Jackie plays with her customized character outside the warehouse, her dice pool options look like this: sneaking (6d6), ranged combat (5d6), bluff (4d6), and close combat (2d6). However, when Jordan plays the game, he chooses a pre-generated character who is a bruiser and thus establishes different values for attributes and skills in the initialization stage. In the IF, his character faces the same choices but a different set of odds: sneaking (2d6), ranged combat (2d6), bluff (3d6), and close combat (8d6). The formula of ability + skill = dice pool is the same, but the likelihood of success for each option changes based on the individual character’s stats.

To this point, I’ve only discussed obstacles that require a skill test to overcome, but the IF writer can add other elements that may have no bearing on their skills but instead help establish the character’s personality and attitude from the onset. For example, a player could choose a value between 1 (low) and 10 (high) for a variety of personality traits: timid to daring, introverted to extroverted, serious to whimsical. Or they could have a pool of points to distribute across the Big Five personality traits: openness, conscientiousness,

extraversion, agreeableness, and neuroticism (Roccas et al. 2002). A third possibility would be to start the game by presenting the player with a series of moral or ethical decisions to make on behalf of their character, such as:

Late at night, you come across a man wearing a suit, slumped, reeking of liquor, sleeping on a park bench. His briefcase is on the ground next to him. Do you:

- (a) Try to rouse the man to consciousness and help him to his feet.
- (b) Call the police and wait for them to arrive.
- (c) Snatch the briefcase and run.

A series of such questions can have answers with different numeric values for different characteristics, and through answering the questions, the player sets the characters' personality traits that could influence options later in the narrative. For example, the option to pickpocket someone may only be visible if the character has an Opportunistic characteristic greater than or equal to 5 and a sneak skill of 7 or greater.

#### 4. Conclusions

The fusion of TTRPG rulesets and IF software like Twine represents a powerful shift in digital pedagogy. By adopting a proven game engine, writers are forced to confront the fundamental questions of narrative choice: not just what a character does, but what they *can do* based on *who they are*, transforming the writing process from a series of static scenes into a dynamic system where choices have consequences for the narrative. This allows students to simulate the complexity of a multi-hour open-world RPG within a single, polished chapter or questline. This process teaches writers to consider what consequential choices should be decided by the player, which in turn should be grounded in the character's identity and personality. Ultimately, whether these students go on to write for major game studios or produce traditional print fiction, this technical framework provides the analytical tools necessary to think deeply about the network of choices that both writers and their characters make in a narrative, and then to think through the logical consequences of their decisions.

**Funding:** This research received no external funding.

**Data Availability Statement:** No new data were created in this study. Data sharing is not applicable to this article.

**Conflicts of Interest:** The author declares no conflict of interest.

#### References

- Aarseth, Espen J. 1997. *Cybertext: Perspectives on Ergodic Literature*. Baltimore: Johns Hopkins University Press.
- Anthropy, Anna. 2012. *Rise of the Videogame Zinesters: How Freaks, Normals, Amateurs, Artists, Dreamers, Dropouts, Queers, Housewives, and People Like You Are Making Art with Games*. New York: Seven Stories Press.
- Freese, Jeremy, Kristi G. Paine, and Christopher J. Case. 2024. *Story Mode: The Creative Writer's Guide to Narrative Design and Game Writing*. New York: Bloomsbury Academic.
- Härenstam, Tomas. n.d. *Year Zero Engine Standard Reference Document v1.0*. Stockholm: Free League Publishing. Available online: <http://freeleaguepublishing.com/wp-content/uploads/2023/11/YZE-Standard-Reference-Document.pdf> (accessed on 7 November 2025).
- Hintze, Nils. 2020. *Vaesen: Nordic Horror Roleplaying*. Stockholm: Fria Ligan.
- Jenkins, Henry. 2006. *Convergence Culture: Where Old and New Media Collide*. New York: New York University Press.
- Montfort, Nick. 2003. *Twisty Little Passages: An Approach to Interactive Fiction*. Cambridge: MIT Press.
- Mulligan, John. 2024. Dublin Firm behind Baldur's Gate Game Posts €249m Profit after Runaway Success. *Indo*, November 22. Available online: <http://www.independent.ie/business/irish/dublin-firm-behind-baldurs-gate-game-posts-249m-profit-after-runaway-success/a2033961431.html> (accessed on 13 August 2025).
- Murray, Janet H. 1997. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. New York: Free Press.

- Open Gaming Foundation. n.d. Open Game License Version 1.0. Available online: <http://opengamingfoundation.org/ogl.html> (accessed on 7 November 2025).
- Reed, Aaron A. 2023. *50 Years of Text Games: From Oregon Trail to AI Dungeon*. Santa Cruz: Changeful World Press.
- Reed, Aaron A., John Murray, and Anastasia Salter. 2020. *Adventure Games: Playing the Outsider*. New York: Bloomsbury Academic.
- Roccas, Sonia, Lilach Sagiv, Shalom H. Schwartz, and Ariel Knafo. 2002. The Big Five Personality Factors and Personal Values. *Personality and Social Psychology Bulletin* 28: 789–801. [CrossRef]
- Salter, Anastasia. 2014. *What Is Your Quest? From Adventure Games to Interactive Books*. Iowa City: University of Iowa Press.
- Salter, Anastasia, and Stuart Moulthrop. 2021. *Twining: Critical and Creative Approaches to Hypertext Narratives*. Amherst: Amherst College Press.
- Schaeffer, Daniel. n.d. *Not Playing Around: Board Games and Intellectual Property Law*. Chicago: American Bar Association. Available online: [http://www.americanbar.org/groups/intellectual\\_property\\_law/resources/landslide/archive/not-playing-around-board-games-intellectual-property-law/](http://www.americanbar.org/groups/intellectual_property_law/resources/landslide/archive/not-playing-around-board-games-intellectual-property-law/) (accessed on 7 November 2025).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.