

# Isofict: Toward Continuous Interactive Fiction

Kaylah Facey<sup>1</sup>, Seth Cooper<sup>1</sup>

<sup>1</sup>Northeastern University, Boston, MA, USA

## Abstract

It is common for interactive fiction to use parser-based interfaces, where users enter commands as free text, with discrete movement and actions. This can both limit the spatial possibilities of a game's world and be an intimidating interface for new players. Thus, we are developing Isofict, a system for interactive fiction that allows the player to move continuously through an updating textual world using a WASD- and mouse-based interface. We present our current progress implementing Isofict as a web-based Inform 7 extension and describe our experience porting two simple games. In the future we hope to compare to more standard parser-based systems.

## Keywords

interactive fiction, video games, Inform 7

## 1. Introduction

Interactive fiction (IF) is an umbrella term for interactive experiences expressed primarily through the use of text, as opposed to graphics. There are many forms of interactive fiction, but parser-based IF, which has an underlying world model affected by typed commands, is the focus of this work [1]. Though parser-based IF has an underlying world model representing a physical space, that model is still generally *discrete*; that is, there is no coordinate system representing relative locations of objects. When a player enters an area, regardless of its conceptual size, they generally have full access to all objects it contains and are unconstrained by distance or occluded visibility. In addition, they immediately lose access to everything in the area they were previously in. This is especially noticeable in games with wide open spaces, where the author must choose between either representing the space as one giant area where the player can access objects that are described as far from each other, or splitting the space into smaller areas where the player loses access to objects that are very nearby. Achieving a more realistic spatial representation requires the game author to write special rules to override the default discrete behaviour [2].

Additionally, parser-based IF is based on a “command line” interface (CLI) where players must figure out what to type, in the way the parser understands, with minimal guidance. While experienced players may be familiar with standard verbs and synonyms used, such an interface can have a high learning curve [3, 4, 5, 6], which may limit IF’s usability for newcomers and reduce the number of players who get to experience IF. Some parser-based IF has eliminated “guess-the-verb” problems by providing a restricted list of verbs [7]. Similarly, text-graphics hybrid games like those created for SCUMM display static images of the game world, which is interacted with by “constructing verbal commands” from a provided list of verbs [8].

Some works have explored ways to represent a discrete world so that it feels more continuous. Seltani [9], an online environment designed as a hypertext multi-user dungeon (MUD) continually updates the description of its discrete world as it changes. Text-based Multiplayer Shooter [10] is a command line MUD where players can in real-time

move around a standard discrete parser-based game map and type “FIRE” to shoot at anyone in the same room as them. Other existing works have experimented with combining a fully rendered 3D world with text. 3DTextAdventure [11] controls a fully rendered 3D world with WASD and parser commands. The unreleased thesis project Text Quest [12] appears to have a similar interaction style but renders its 3D world using the names of objects as their textures. The academic game Façade [13] also renders a 3D world that players interact with through arrow keys, mouse clicks, and typed text. None of these has an available authoring system, and their 3D worlds have to be precise enough for rendering, a not-insignificant authoring burden.

To attempt to expand the spatial possibilities of IF without requiring full 3D rendering, we are developing an authoring system for *continuous interactive fiction*, where the underlying world model is represented on a 2D coordinate plane with simple polygons. We are adapting Inform 7 to take advantage of its popularity and detailed underlying world model. We have developed the *Isofict* Inform 7 extension and JavaScript library. The name was chosen for the term “isovist”, which describes the visible region from a point in space [14]. We describe some of our progress on our technical approach to developing the system and its interface, and our experience “porting” two simple Inform 7 games to *Isofict*. We find that for these two games, very little modification is required to the Inform 7 source code to support the new spatial model. In the future, we hope to have user studies of both the playing and authoring interfaces of *Isofict*. We aim to release *Isofict* to the Inform 7 community, along with an original game designed from scratch to best showcase *Isofict*’s capabilities.

## 2. System Overview

The *Isofict* system uses an underlying 2D spatial model, which allows the player to move continuously throughout the world (currently using WASD to walk and turn). As the player moves, the description of the world is continually updated, based on where the player is looking and what they can see (similarly to Seltani [9]). Players interact with objects by clicking on hyperlinks. In addition to the text output, a simple 2D top-down map, including the player and their fields of vision, is included.

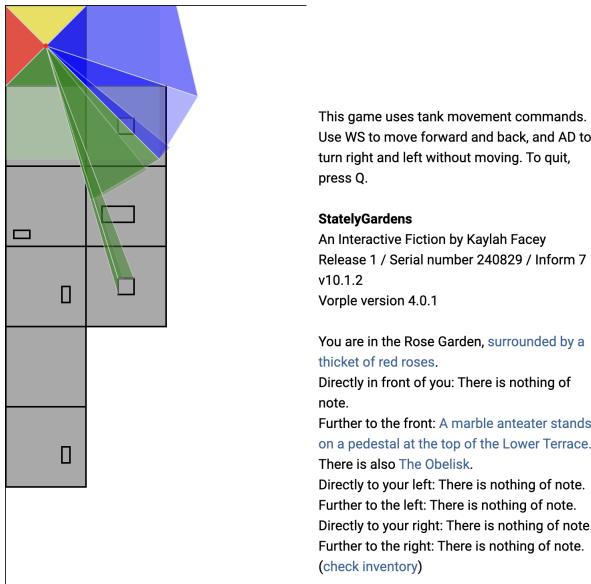
*Isofict* is composed of an Inform 7 extension paired with a JavaScript (JS) library. To integrate Inform 7 and JS, we use *Vorpyle* [15], an Inform 7 interpreter that provides a translation layer API between Inform 7 and JS. *Vorpyle* also provides

AIIDE Workshop on Intelligent Narrative Technologies, November 18, 2024, University of Kentucky Lexington, KY, USA

✉ facey.k@northeastern.edu (K. Facey); se.cooper@northeastern.edu (S. Cooper)

>ID 0009-0001-1787-5562 (K. Facey); 0000-0003-4504-0877 (S. Cooper)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



**Figure 1:** *Stately Gardens* starting map and text. The 2D world model shows the coordinate representation of the game world. The POV is the red dot in the upper left. The coloured triangles show the visibility to the front (green), right (red), and left (blue). The area behind the POV (yellow) is not visible. The visible range extends infinitely, but the visualization is cut off at 200 pixels if there is not a visible object further than that, such as the Obelisk reached by the longest green triangles.

HTML for rendering the game text and does not permit accessing the game text directly. To change the display of the game, it is necessary to interact with the rendered HTML rather than with *Vorpel* directly. The *Isofict* library and extension are designed to work well with a primarily mouse- and WASD-based interface, and we have created one such JS front-end.

To update an Inform 7 game to use *Isofict*, the game author must provide a JS file containing a JSON object that defines the starting coordinates, polygon shapes, and heights of game objects. It must also have the starting coordinates of the player (“POV”) and the directions they can look in. In addition, the Inform 7 code must be annotated with *Isofict* keywords (see code with the case study *Cloak of Darkness*).

## 2.1. JavaScript Library

The JS library for *Isofict* provides functions for calculating the visibility of 2D polygons plus height. Visibility is calculated using the 2D raycasting algorithm described and implemented in Haxe 3 by Red Blob Games [16]. We have modified the code of the GitHub repository “2d-visibility” by David Neilsen [17], which is a TypeScript fork of “2d-visibility” by Cyril Silverman [18], itself a JS port of the original Haxe 3. Our main changes are to support custom view ranges (rather than 360° vision), and to add the 3D visibility heuristic that objects may be occluded by anything of equal height or taller. We chose to modify a 2D algorithm over using true 3D visibility for simplicity of implementation. The POV is a 2D point in space (without height), with an angle of orientation and configurable fields of vision relative to the orientation. For example, the “forward” field of vision may be a 90° wedge centered on the angle of orientation. Objects that are visible in multiple directions

are only marked visible in the direction that includes the greatest percentage of their edges. Each time the POV or orientation changes, the visibility is recalculated.

## 2.2. Inform 7 Extension

The *Isofict* extension provides helper functions that interface with the JS. In addition, the extension overrides the Inform 7 default world model to use the JS coordinate model instead. Inform 7 authors can use the functions provided by the *Isofict* extension rather than invoking JavaScript commands directly.

The Inform 7 extension is designed to work with a mouse. For example, descriptions of game objects include clickable links to commands like “EXAMINE” and “TAKE”.

## 2.3. JavaScript Front-End

We have implemented a simple JS front-end for *Isofict* using the Svelte framework [19]. The front-end hides *Vorpel*’s mandatory text input box, requiring interaction through WASD and mouse only. Hyperlinks are natively supported by *Vorpel*, and WASD is enabled by passing *Isofict* commands to Inform 7 via *Vorpel*.

## 3. Case Studies in Developing with Isofict

To further refine *Isofict* and evaluate its usability, we have modified two existing Inform 7 documentation games. Both games use viewing angles for “forward”, “left”, and “right” that each take up 90°.

### 3.1. Stately Gardens

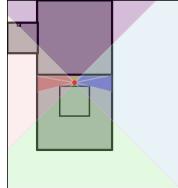
As a proof-of-concept for *Isofict*, we re-implemented the Inform 7 documentation example game *Stately Gardens* [2], a game about exploring a garden. *Stately Gardens* was originally designed to demonstrate Inform 7 workarounds for handling wide open spaces.

In *Stately Gardens*, the player is free to explore an outdoor garden divided into multiple Inform 7 “rooms”. A system of custom Inform 7 rules makes large objects viewable from outside the room they are in. The player can walk around and look at things, but there is no objective or win condition.

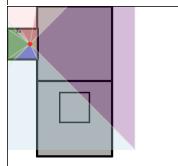
We have rewritten *Stately Gardens* from the ground up to use *Isofict*. Using *Isofict*, we were able to exclude the special code for viewing objects outside of the current room. For simplicity, we omitted the few inventory objects. In the resulting game, the player can move around the *Stately Gardens* map, and visibility information is provided by *Isofict*’s JavaScript calculation instead of a system of Inform 7 rules around sizes and distances. See Figure 1 for the beginning of the game.

### 3.2. Cloak of Darkness

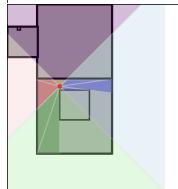
For our second case study, we wanted to evaluate how many changes are needed to existing Inform 7 code to make it compatible with *Isofict*. The game we chose to adapt is *Cloak of Darkness* [20] (*CoD*), which is historically a game that has been implemented in most new IF programming languages and tools as the IF “Hello World” [21].



You take a step forward.  
You are in the dark.  
Directly in front of you: There is nothing of note.  
Further to the front: There is nothing of note.  
Directly to your left: There is nothing of note.  
Further to the left: There is nothing of note.  
Directly to your right: There is nothing of note.  
Further to the right: There is Foyer of the Opera House.  
Further to the right: There is nothing of note.  
(check inventory)  
Blundering around in the dark isn't a good idea!



You turn to the left.  
You are in the Cloakroom.  
Directly in front of you: There is nothing of note.  
Further to the front: There is nothing of note.  
Directly to your left: There is nothing of note.  
Further to the left: There is nothing of note.  
Directly to your right: There is Foyer of the Opera House, and the small brass hook (on which is a velvet cloak (take)).  
Further to the right: There is nothing of note.  
(check inventory)



You take a step forward.  
You are in the Foyer Bar.  
Directly in front of you: There is nothing of note.  
Further to the front: There is nothing of note.  
Directly to your left: There is Foyer of the Opera House, and a scrawled message.  
Further to the left: There is nothing of note.  
Directly to your right: There is nothing of note.  
Further to the right: There is nothing of note.  
(check inventory)  
The message has been carelessly trampled, making it difficult to read. You can just distinguish the words...

\*\*\* You have lost \*\*\*

In that game you scored 1 out of a possible 2, in 50 turns.

Would you like to RESTART, RESTORE a saved game, QUIT or UNDO the last command?

**Figure 2:** Screenshots of key *Cloak of Darkness* moments: the dark Bar, the Cloakroom (after hanging the cloak), and the ending (after attempting to read the trampled message).

In *CoD*, the player is able to walk around an opera house consisting of three rooms. The “Bar” is initially dark, and if the player stays too long in it, they damage a message on the floor. When the black cloak the player is wearing is left in the “Cloakroom”, the Bar becomes lit, and the player can see the message. The game is won if the message is not damaged.

To adapt *CoD*, we again used the version from the Inform 7 documentation [22], but in this case rather than rewriting the game from the ground up, we modified the existing code as little as possible. We found that very little game-specific modification was required, and the result is a fully playable implementation of *CoD* using a coordinate system. While modifying *CoD*, we further refined *Isofict*, as *CoD* makes use of some Inform 7 default features that were not previously implemented (namely, visibility in dark rooms and the use of portable objects). We took advantage of the new world model by changing the logic of the message in the Bar such that the player must actually walk over it (using WASD) to damage it. See Figure 2 for the key moments in a losing game.

The following code illustrates typical code changes required to port *CoD* to *Isofict*. Code removals are tagged with ‘-’, and additions are tagged with ‘+’. Unchanged code is

left untagged. Ellipses represent code omitted from this comparison. Comments are in square brackets.

```
[Inform 7 rooms with direction-based connections are replaced by
special "geo-rooms" which have coordinates defined in JSON.]  
- The Bar is south of the Foyer.  
+ The Bar is a geo-room in the Opera House.  
...  
[Similarly, objects must be tagged as "geometric" to be recognized by
the Isofict extension. The message is now noted as a supporter
to recognize that the player can now walk over it.]  
- The scrawled message is scenery in the Bar.  
+ The scrawled message is a geometric scenery supporter in the Bar.  
...  
[In the original game, the message is damaged by any action in the
Bar other than leaving it. Now we check each time the player
changes coordinates ("geo-moving") whether they have stepped
on the message.]  
- Instead of doing something other than going in the bar when in
darkness:  
+ After geo-moving:  
    [To check if the player is within a given object's coordinates,
    use the function "geometrically enclosed by"]  
    + if the player is geometrically enclosed by the Bar and the
    Bar is geo-dark:  
        + say "Blundering around in the dark isn't a good idea!"  
    + if the player is geometrically enclosed by the message:  
        if the message is not trampled, now the neatness of the
        message is the neatness after the neatness of the
        message;  
        - say "In the dark? You could easily disturb something."  
        [Now, the player can scuff the message even when the Bar
        is lit, so we add a message for that.]  
        + if the Bar is not geo-dark:  
            + say "You've gone and scuffed the message on
            the floor.[if the message is trampled]You doubt
            you'll be able to read it now."  
    - Instead of going nowhere from the bar when in darkness:  
        - now the message is trampled;  
        - say "Blundering around in the dark isn't a good idea!"  
["Updating the isovist" refers to updating the visibility of Inform
7 objects based on changes to the JS model.]  
+ Instead of doing something other than geo-moving or geo-turning or
looking or updating the isovist when the player is in shadow:  
+ say "In the dark? You could easily disturb something."
```

## 4. Discussion

We are encouraged by the relatively little game-specific code required to port an existing game to *Isofict*. After defining game objects in the JavaScript coordinate system, an author mainly only needs to do basic tagging of their Inform 7 code to allow it to use the extension. In the future, we will remove the requirement to define game objects in JavaScript by providing Inform 7 commands to create objects in the coordinate system. In addition, we intend to refine *Isofict* parser commands to facilitate including the option to toggle between mouse-based input and command line input to support users’ preferences or accessibility needs [3].

The current version of *Isofict* does not yet support all features of Inform 7. We are continuing to improve the handling of portable items, as well as support for supporters, containers, and objects that form a part of other objects.

There are also some particularly notable limitations to the coordinate system. Most obviously, it doesn’t support true 3D visibility, which is the most disruptive in situations involving small objects. For instance, if multiple small objects of the same height are in a row along the player’s orientation angle, only the front-most will be visible, even when the game author might expect the player to be able to see all of them. Similarly, there is not currently a good way to represent objects on top of other objects or objects that “float”, such as tabletops. Objects with a height occlude the entire area under them. To handle these cases, *Isofict*’s visibility system could be improved with more sophisticated height handling heuristics, or it could transition to true 3D raycasting for simple 3D objects.

Another limitation of *Isofict* is the field of view system, where objects are only marked visible in one direction. In practice, this can cause confusion when objects are placed into non-intuitive directions. For example, when facing an object that is very close, its front view contains a much smaller part of the visible object than is visible in the periphery. Though a player would expect it to be placed in the “forward” view, it is instead added to the “right” or “left”. To handle this, *Isofict* could incorporate information on how close objects are when making visibility calculations.

Development of *Isofict* has often been hindered by the need to override the default behaviours of Inform 7 and the *Vorple* interpreter. Inform 7’s rules for when objects are and are not available (“scoping” and “reachability” rules) are not easily surfaced or replaced. *Vorple*, on the other hand, directly updates HTML, making it challenging to replace the front-end.

Although we initially intended to make the player interface text-only, during development we found that it could be difficult to play the game without the 2D map. However, we would like to improve *Isofict* such that the text description of the world would be enough for the player to navigate.

We would like to investigate the effect of *Isofict* on the player experience. We hope that players will find the changed visibility system more immersive without being frustrating. We would also like to explore the impact of including the 2D overhead map, or not, on the player experience.

We also intend to investigate *Isofict*’s effect on game authoring. We hope to see entirely new spatial mechanics that form the basis of games with fine-grained coordinate models, but we also hypothesize that for most games, it does not make sense to create a coordinate model for every object. We look forward to seeing *Isofict* used in a hybrid manner to create spatial sections of games or to address movement or visibility needs in a coarse-grained fashion. For example, most objects could be part of discrete rooms as usual, while the visibility of landmarks could be addressed via the visibility system.

## 5. Conclusion

Some interactive fiction is designed to represent an underlying 3D world; however, existing tools like Inform 7 support a more discrete than continuous spatial representation. To enable more realistic visibility where there are wide open spaces or objects that can be interacted with from other rooms, game authors must write ad-hoc rules to handle their use cases.

To address the desire for more continuous spaces and realistic visibility in parser-based IF, we have developed an Inform 7 extension and JavaScript library, *Isofict*, that provides a coordinate-based world model and visibility system that allows Inform 7 games to request the visibility of objects in the coordinate system based on the location and orientation of the player. After using *Isofict* to modify two existing Inform 7 example games as case studies, we find that little game-specific code is needed.

We plan to conduct a user study to get feedback on the experience of playing an Inform 7 game that uses *Isofict*. In addition, we intend to refine the *Isofict* extension to even more easily integrate into existing Inform 7 games. We also plan to improve the visibility heuristics and 3D processing. To showcase the capabilities of *Isofict*, in addition to the

case studies described here, we will release an original game using it. We will then release *Isofict* as a public extension and solicit feedback from game authors on how well it serves their needs.

## Acknowledgements

We would like to thank Chris Martens for their contribution to the ideation of the design of this system.

## References

- [1] IFWiki, Parser-based interactive fiction, [https://www.ifwiki.org/Parser-based\\_interactive\\_fiction](https://www.ifwiki.org/Parser-based_interactive_fiction), 2022. Accessed: 2024-08-30.
- [2] G. Nelson, *Stately Gardens*, Game [Inform 7], 2022. URL: [https://ganelson.github.io/inform-website/book/WI\\_18\\_29.html](https://ganelson.github.io/inform-website/book/WI_18_29.html), accessed: 2024-08-30.
- [3] M. J. Heron, A case study into the accessibility of text-parser based interaction, in: Proceedings of the 7th ACM SIGCHI symposium on engineering interactive computing systems, 2015, pp. 74–83.
- [4] A. Feizi, C. Y. Wong, Usability of user interface styles for learning a graphical software application, in: 2012 International Conference on Computer Information Science (ICCIS), volume 2, 2012, pp. 1089–1094.
- [5] T. Fellmann, M. Kavakli, et al., A command line interface versus a graphical user interface in coding vr systems, in: Proceedings of Second IASTED International Conference on Human Computer Interaction, 2007.
- [6] E. Dillon, M. Anderson-Herzog, M. Brown, Studying the novice’s perception of visual vs. command line programming tools in cs1, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, volume 56, SAGE Publications Sage CA: Los Angeles, CA, 2012, pp. 605–609.
- [7] E. Short, If only: Text adventures for people who hate guessing the verb, Rock Paper Shotgun (2016). URL: <https://www.rockpapershotgun.com/if-only-text-adventures-for-people-who-hate-guessing-the-verb>, accessed: 2024-08-30.
- [8] M. L. Black, Narrative and spatial form in digital media: A platform study of the scumm engine and ron gilbert’s the secret of monkey island, Games and Culture 7 (2012) 209–237.
- [9] A. Plotkin, *Seltani*, Game [Web], 2015. URL: <https://seltani.net>, accessed: 2024-08-30.
- [10] E. Lenk, *Text-based Multiplayer Shooter*, Game [Web], 2012. URL: <http://eigen.pri.ee/shooter/>, accessed: 2024-08-30.
- [11] auspaco, *3DTextAdventure*, Game [PC], 2014. URL: <https://www.indiedb.com/games/3dtextadventure>, accessed: 2024-08-30.
- [12] S. Robinson, Grad student’s thesis, text quest, headed to steam, The Media School (2015). URL: <https://mediashool.indiana.edu/news-events/news/item.html?n=grad-students-thesis-text-quest-headed-to-steam>, accessed: 2024-08-30.
- [13] M. Mateas, A. Stern, Façade: An experiment in building a fully-realized interactive drama, in: Game developers conference, volume 2, Citeseer, 2003, pp. 4–8.
- [14] A. Turner, M. Doxa, D. O’Sullivan, A. Penn, From isovists to visibility graphs: a methodology for the

- analysis of architectural space, Environment and Planning B: Planning and design 28 (2001) 103–121.
- [15] J. Leinonen, Vorple, <https://vorple-if.com>, 2012. Accessed: 2024-08-30.
  - [16] R. B. Games, 2d visibility, <https://www.redblobgames.com/articles/visibility/>, 2020. Accessed: 2024-08-30.
  - [17] D. Neilsen, Petah/2d-visibility, <https://github.com/Petah/2d-visibility>, 2019. Accessed: 2024-08-30.
  - [18] C. Silverman, Silverwolf90/2d-visibility, <https://github.com/Silverwolf90/2d-visibility>, 2016. Accessed: 2024-08-30.
  - [19] R. Harris, Svelte, <https://svelte.dev/>, 2016. Accessed: 2024-08-30.
  - [20] R. Firth, The “Cloak of Darkness” specification, <https://web.archive.org/web/20190916101552/http://www.firthworks.com/roger/cloak/>, 2002. Accessed: 2024-08-30.
  - [21] IFWiki, Cloak of Darkness, [https://www.ifwiki.org/Cloak\\_of\\_Darkness](https://www.ifwiki.org/Cloak_of_Darkness), 2022. Accessed: 2024-08-30.
  - [22] G. Nelson, *Cloak of Darkness*, Game [Inform 7], 2022. URL: [https://github.com/ganelson/inform-website/blob/main/docs/book/WI\\_17\\_2.html](https://github.com/ganelson/inform-website/blob/main/docs/book/WI_17_2.html), accessed: 2024-08-30.