6-15-2023

Wyclef: A Rational Reconstruction of Versu Social Engine

James Dameris

Rosaura Hemandez

SANTA CLARA UNIVERSITY

Department of Computer Science and Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

James Dameris, Rosaura Hernandez

ENTITLED

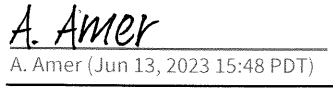
Wyclef: A Rational Reconstruction of Versu Social Engine

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE AND ENGINEERING

2023-06-13

Thesis Advisor(s) (use separate line for each advisor) date



Jun 13, 2023

Department Chair(s) (use separate line for each chair) date

WYCLEF: A RATIONAL RECONSTRUCTION OF VERSU SOCIAL ENGINE

By

James Dameris, Rosaura Hernandez

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Science and Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Computer Science and Engineering

Santa Clara, California

2023

Wyclef: A Rational Reconstruction of Versu Social Engine

James Dameris, Rosaura Hernandez

Department of Computer Science and Engineering

Santa Clara University

June 15, 2023

ABSTRACT

Video games often lack an accurate representation of the person playing the game, causing a disconnect between the player and the character's story. With Versu, a proprietary social simulation engine, not only did choices become more connected to the player, but the player could choose to take the place of any of the characters present in the story. This allows the player to find themselves in the story, while also giving vast replayability to the game. Wyclef is an open source, rational reconstruction of Versu providing increased authoring speed of stories and availability of the social engine to more people. We found that based on the limited documentation provided, it is still very possible to fully rebuild Versu. Additional work on providing authoring of characters and worlds to be parsed by the engine could further flesh out the capabilities of Wyclef.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our Advisor, Dr. Max Kreminski, for their guidance, support, and encouragement throughout the research and design process. In addition we would like to thank the School of Engineering for providing resources for working on our project.

Table of Contents

List of Figures

1. Introduction

Our team's senior design project, Wyclef, is an open-source rational reconstruction of Versu, a tool that will be described later in this paper. In the following sections, we will detail the motivations behind creating the project, and a general overview of the steps we took to get the project to where it is today.

1.1. Problem Statement

Video games have advanced drastically over the last 20 years, from pixelated point-and-clickers to full-fledged immersive stories. Many modern video game worlds are filled with hundreds of non-playable characters (NPCs), and despite them looking more visually appealing, they can still feel uncanny due to their lack of personality, background, and engagement with the player. When engaging players they often repeat the same lines and are oblivious to the events of the world around them. Games like The Sims look into solving this issue but with simple emotional responses rather than words. More complex solutions have been derived like that of the Versu Social Simulation Engine, but they are locked behind a company’s copyright, with the only knowledge of its workings lying within published documentation. Using JavaScript, we will be reconstructing an open-source version of Versu, with a focus on its social practice engine. Using JavaScript allows wide accessibility to developers all around and the ability to quickly implement a frontend to test the social practice engine. Reconstruction of Versu will be done through architectures described within research papers by the creators of Versu and by extrapolating off of adjacent works such as Ensemble. This open-source engine will give users an authoring tool to ease the implementation of social practices into games and expose technology that has been lost, allowing others to further the technology in future projects.

1.2. Project Purpose

The purpose of our project is to help video game developers improve user experience by creating a more diverse gameplay whose purpose is to make the user feel like they are the narrator to their own story. Our team believes that the gaming industry has a long way to go when it comes to creating a more engaging storyline for their games. To accomplish this goal, we wanted a tool that enforces plausible actions depending on the social practice. This would allow us to attain a storyline that didn't completely change the objective of the story while still giving the user autonomy to take the adventure in the direction they would like. Another attractive attribute of Versu is its Non-Playable Characters that had backstories of their own which gave them personalities and goals of their own outside of the user's own story. This made them seem less like background characters in the user's escapade and more like people that are on their own journey. The social practice that the user would be able to play in is provided by the developer using Wyclef to allow them to create the stage that they would like their users to find themselves in. These qualities, in conjunction with the fact that Versu is no longer available for the public anymore, solidified our team's choice to recreate Versu.

1.3. Project Proposal

Our proposal is a rational reconstruction of Veru using any documentation left behind about the technology. To make this project more approachable, we deconstructed the project into parts; the social practice parser, the logic database, the action executioner, the world initialization file, and the character file. To ensure that we would have something to present by the end of the year, we prioritized the social practice parser, the logic database, and the action executioner which we were able to get working.

1.4. Background Research

Background research for the reconstruction of Versu was the most essential step of the process as we were reconstructing Versu from documentation and journals alone. This meant gathering a firm grasp on design decisions and extrapolating on work not shown in the literature were required.

1.4.1. Versu

As stated before, this is a rational reconstruction, which means we will be using the documentation of Versu as a guide for our entire design rationale. Versu is an “interactive narrative platform” (Short, 2013): in other words, a game engine that enables people to easily build interactive narratives. Building such a game engine takes a lot of interweaving systems. Versu’s creators clearly documented an architecture (figure 1) for the engine, with its core being a database.

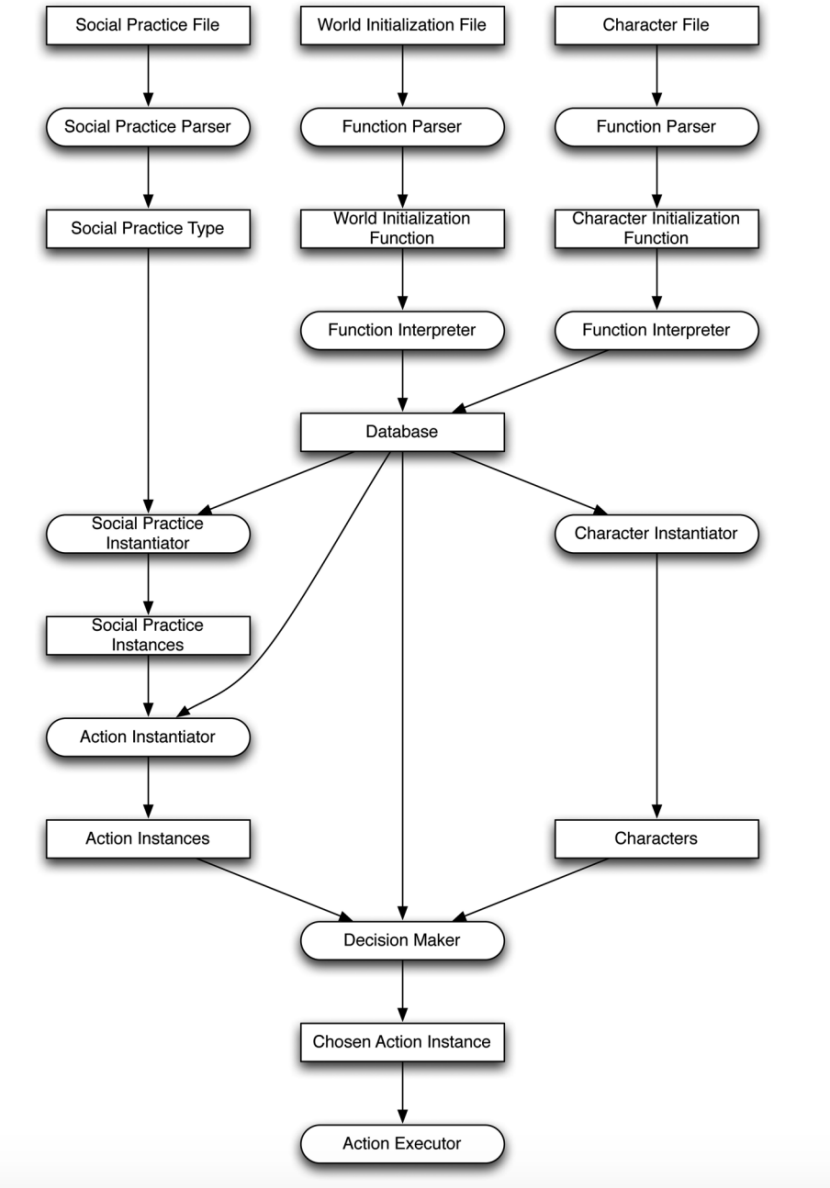


Figure 1: Versu's Architecture

The social practice files, the world files, and the character files build up the database. Social practices "describe a type of recurring social situation" (Short, 2014), as described by Emily Short, which describes what the characters in that situation can do, and not what they should do. The difference between can do and should do is an important

delimiter as Versu uses a relaxed drama manager, giving more power to the individuals rather than dictating what each character should do given a situation. The character files are exactly what one would assume, describing character specific information. Short describes the world as “Everything that is the case”[2], implying that the world file describes things of a more static nature, the normalities of the world the characters find themselves in. The database itself doesn’t inherently understand the contents of the different files. It requires a parser for each of the different files as well as a domain specific language (DSL) called Praxis. A DSL enables the structure of the database to be uniform, allowing for the processing of the database to be as simple as processing a set of sentences. The database describes the initial state of the world after retrieving the initial information from the files. Practices and characters are then instantiated, and actions from the practices are instantiated based on the characters interacting within the specific social practices. Once the actions have been instantiated, the database, action instances, and characters are fed into the decision maker.

1.4.2. Other social simulation frameworks

We looked into other social simulation frameworks to help with the interpolation process of rational reconstruction, particularly with the game Facade’s social simulation engine. Facade, compared to Versu, uses a drama manager which, “looks at the current state of the story and the characters and decides which unused beats would help move the story through the intended tension arc.” (Thompson, 2020) By opting for this type of drama manager, Facade’s engine gives the power to drama manager, compared to Versu which doesn’t use beats and prefers to give the actors the power to choose from what they can do, rather than what is dictated to be done by the drama manager. In addition to Facade, we took Comme il Faut, abbreviated CiF, as another Social Simulation engine to help with the extrapolation process of building Wyclef. CiF’s architecture is quite similar to the one documented by Versu due to it feeding a central procedure with information like social state and characters. Between Facade’s engine and CiF, we were able to fill in some of the gaps in the Versu documentation.

2. General Description of the Project

At the start of this project, we set out to make guidelines for ourselves on what aspects of the project we wanted to prioritize finishing first. We prioritized creating the social practice parser because the user interacts with Wyclef by firstly giving a social practice JSON file. In addition, we also prioritized creating the database to be able to instantiate the file into a social practice and actions. To be able to see that these two parts were working, we created a frontend that had instantiated social practices and actions for the user to interact with. The elements that we didn't prioritize and didn't get to were the world instantiation file that would have limited a character's action according to their locations. An example of this would have been that a character at the bar wouldn't be able to have a conversation with a character at the park unless they were on a phone call. Another element that we didn't get the chance to implement was the character file, where we would have further developed the ways that the user would be able to customize their characters. Further details about the requirements we created and the constraints we found while working on the project are listed below.

2.1. Functional Requirements

Onto the functional requirements of the project, or our implementation goals.Firstly, we wanted to provide a framework for the user to become a part of the story. To complete this requirement, we set out to create a Drama Manager that would suggest what actions can be done instead of enforcing what must be done. When taking an action, as long as the preconditions for said action persist, we let them repeat the same actions over and over until the preconditions are no longer met. Actions like drinking a beverage, will remove the ability to repeat the action since they will no longer have a beverage to drink. Next, when given a JSON file, we wanted to be able to parse this file to extract a social practice definition. We were able to successfully create a parser that parsed JSON files and sent the actions from the file to the front-end where users were able to use the actions

from the file given on their characters. In addition, another component we wanted to have is a database to store the state of instantiated social practices and actions. This database would hold information like the current state of characters, which might affect what actions they were able to perform next, or other information about characters like the roles they were given in different practices. Finally, we wanted a user interface to test the Social Sim Engine which we did by implementing a front-end interface that allows the user to create characters and choose the actions they would like to see happen for their own personal story.

2.2. Non-Functional Requirements

Now onto the non-functional requirements. First on the list was to provide reasonable actions given the social situation. We wanted to avoid situations in which characters performed seemingly random actions, like making a smoothie in the middle of a business meeting. While we want every user's story to be unique, we want to ensure that every action fits into the social practice(s) that are currently in place. Also, we wanted the user to feel like they were writing the story instead of an audience member to an already scripted story. We want the feeling of an open world story for the user to not feel restricted on what will make the story progress and instead be curious about the adventures that await them with each action they take.

2.3. Design Constraints

Some constraints that came up as we started our project is the limited information about Versu. Versu was only described through papers and a very limited amount of its architecture was documented which made this project a logical reconstruction. We spent multiple hours trying to imagine how to start recreating something only described in words so we had to stray from the original blueprint when we saw fit. In addition, it was a

great help to also reference other social simulators to gain a more clear understanding on the kind of project we set out to do.

3. User Stories

To outline some of the scenarios we expect Wyclef to be used under, we will provide three user stories. What we want to communicate with the user stories is that Wyclef is an easily accessible open-source project that can be used by people with varying degrees of expertise in coding. Our project can also either be used as it is or changed depending on the needs of the developer. Considering that the developer will add their own JSON files, they are able to add actions to their practice file that they believe to be logical under different circumstances to ensure that the story will go in the right direction without hovering over the user trying to enforce things.

3.1. User Persona (serious)

Tony is a 40-year-old senior in the gaming industry. When first starting in the industry he spent his time researching social simulators like Versu as a basis for how he wanted to create his games. Unfortunately, as years went on, the tool he was looking forward to using was never released to the public and remained proprietary. He is trying to look for similar tools, but they left very little documentation behind so he has very little hope of finding a new tool that uses Versu as its basis. That’s when he stumbles upon Wyclef, a tool that closely resembles Versu but is not fully fleshed out. This great discovery, has motivated him to invest his time on further developing Wyclef to become a tool that is as close as Versu as possible so his time spent researching would not go to waste. Once satisfied he will be able to use this tool in any future games he will develop and share it with others in his company for others to use also.

3.2. User Persona (individual free time)

Bryann is a 19-year-old gamer who just finished the 4th, and final game from his favorite series that he's been playing since he was 12, but has been left dissatisfied with the ending. This series is his whole childhood, he spends countless hours beating each game and would look forward to the next. With this concluding game to the series, he believes that he would be able to end this chapter of his life content but doesn't feel that way. He feels that the ending is uncharacteristic of the plot and rushed, so he ponders how he could make his favorite game end the way he wants. That's when he stumbles upon Wyclef, a tool where he can simulate the ending scene that he thinks the game deserved. So he creates a JSON practice file that has all the other endings that would be 1000% better than what was in the game. He shares the Wyclef website that has his JSON file with his friends so they can also end the game the way they want. With these worries set aside, they are now ready to embark on a new journey starting a new game.

3.3. User Persona (indie game studio)

Samantha is a 18-year-old fresh college graduate that has always dreamt of working at a gaming studio since she was young and first fell in love with the gaming industry. Throughout college, Samantha had been working on individual projects which impressed many companies and was able to land a job at her dream company. Now she is taking her first steps into the industry and is working on the story production of the new game the company wants to release. To not waste resources by getting started with the animations before the story is fully fleshed out, Samantha wants to test out all the possibilities on how users would use the wide range of actions available to them. To test that the available actions are being used in the way she intended them, she creates JSON files with all the actions she wants to see being taken in the game. She also uses the Wyclef website to put something in front of her test users to see how they would choose to unfold their journey in the game. Samantha is able to get feedback from her test subjects on

whether the set of actions is diverse enough, interesting enough, and if they make sense. Thanks to Wyclef, Samantha can edit her story using the feedback and improve it to take a step further to getting the game out to the public.

4. Application Architecture

As with most web-based applications, we built out a frontend and a backend for Wyclef. The backend was the major focus of the project, but the frontend proved to be extremely useful for purposes such as debugging and authoring.

4.1. Frontend

4.1.1. UI Thoughts and Reasonings

For the frontend of our project, we decided that we wanted to go with a minimalistic design that resembles a book or a script. To get this effect we chose to only use black and white elements and have the screen split into two like an open book. On the right side of the screen,we have a list of actions that each character takes as the user chooses them, when a new action is taken the character name and the latest action is animated onto the screen in a typewriter fashion to make it seem like the user is writing the story as it unfolds. On the left side of the screen is the section that the user interacts with, when the screen is first displayed this is where we have an input field for the user to add the names of the characters they want to create. Once finished, the users press a button that states they are done adding their characters.This is when the input field is removed from the screen as it is no longer needed. To make it clear which character’s turn it is, we have a list of the character names on the bottom of the screen, with the current character’s name bolded. Following, this section is filled with a list of buttons of actions that the user can take for the character they are currently playing as. In the future, we want the buttons to be replaced with an input field where their typed action is matched to the closest action

available for the social practice. With the minimalist design in addition to the animations we want our frontend to be straightforward while also being interactive.

4.1.2. UI Screenshots

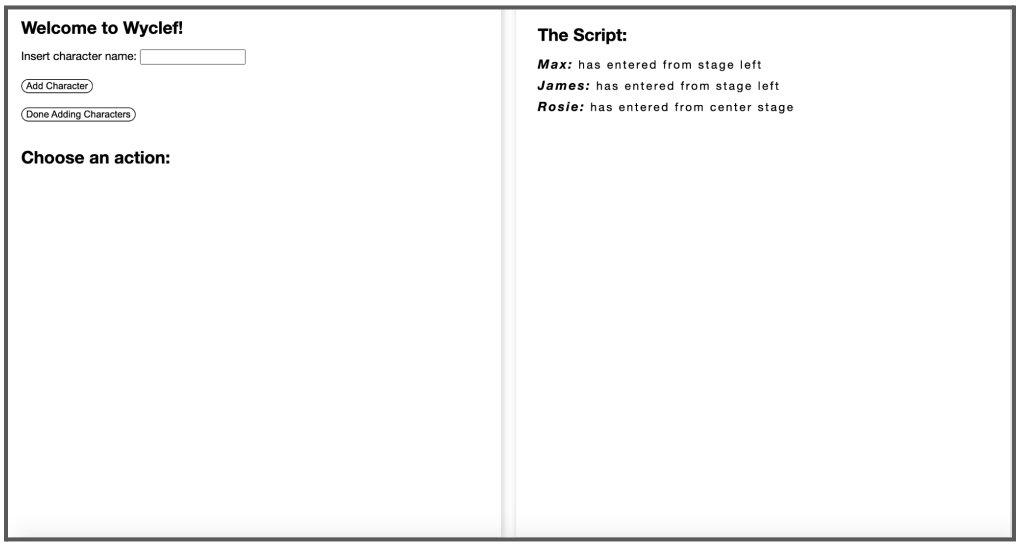


Figure 2: Frontend: Adding characters

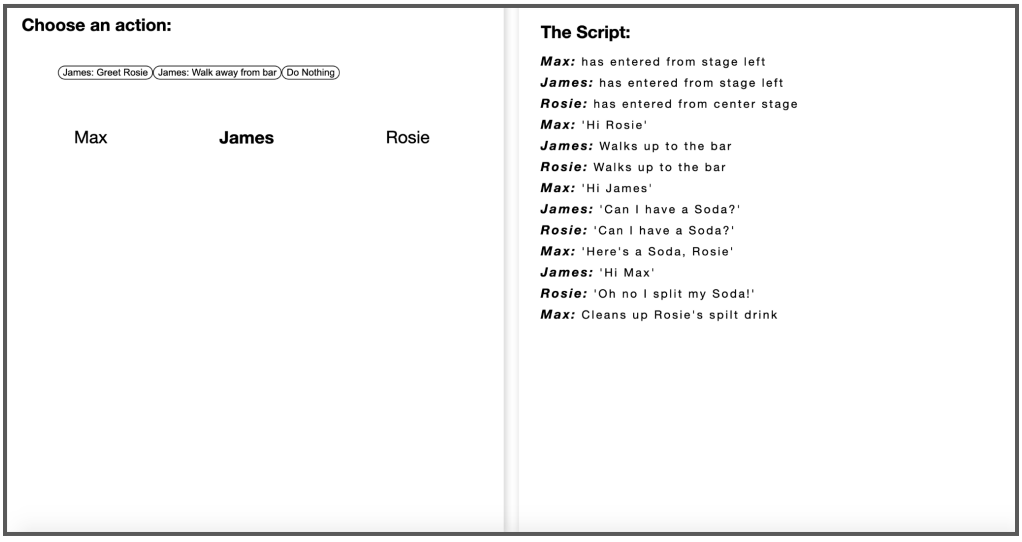


Figure 3: Frontend: Picking actions

4.1.3. Implementation

We used a combination of HTML, CSS, and JavaScript to create the frontend of the project. These languages best fit our needs for a simple interface to help us debug our parser and database. We chose these languages because we had prior experience using them and because they are widely used, which will make it easier for anyone using our project in the future to edit it.

To construct the right side of the display, whenever a new character is added or a new action is taken, the display animates the character name next to their action next in line. This action string has been changed to be human readable to make it more interesting to read and the typewriter style has been provided in our CSS style. Now onto the left side of the display, when a character is created, their name is added onto an array to keep track of which character's turn it is. Once all characters have been created, the list of actions are taken from the social practice and their text put onto buttons for the user to choose from. When an action is chosen, this is communicated back to the database and the next character is picked to take an action next.

4.2. Backend

The engine of Wyclef is built in JavaScript, utilizing JavaScript Object Notation (JSON) as a way to describe social practices so that they can easily be parsed. We decided on JavaScript due to it being the most used programming language in the modern day (Berkeley, 2014) which allows for more users to be able to interact with Wyclef.

4.2.1. Practice Layer

The practice layer consists of social practices that match the template (figure 4).

```
const template = {
  id: "template",
  name: "[Role] doing something",
  data: ["optional list of static data regardless of practice instance"],
  init: ["list of data specific to this practice instance to be initialized"],
  roles: ["list of roles"], // must be start with uppercase for each role
  actions: [
    {
      name: "[Actor]: Do something",
      human_readable: "Does something (an actual sentence",
      conditions: [
        "list of conditions"
      ],
      outcomes: [
        "list of outcomes"
      ]
    }
  ]
}
```

Figure 4: Template for Social Practices

In a social practice object there are four primary attributes: id, name, roles, actions. The id contains a short string descriptor of the practice which is used for the purpose of the database. The name is a sentence description of the practice, like "[Greeter] is greeting [Greeted]". Square-bracketed variable names, like "[Greeter]", are then substituted in instances of this practice by the name of the character bound to the appropriate role. Roles describes the roles actors can take within the practice. For instance, in a bartending practice, there is one role, the bartender, whom the practice belongs to, but there are many actors who interact with the practice. These other actors without a role are the customers who can interact with the practice but are not required for the practice to exist, as a bartender can be without customers.

Actions are more complex and contain four of its own attributes: name, human readable, conditions, and outcomes. Name is the name of the action being taken by the Actor. Human readable is an actual sentence that describes the action such as "Bartender serves Customer a Beverage," where capitalized variable names, like "Bartender" and

“Customer,” and “Beverage” are then substituted in instances of this practice by the name of the character or data bound to the appropriate variable. Conditions is a list of Praxis-like sentences describing the state of the database required to take such action, for instance checking that the customer has ordered a drink before the bartender can fulfill their order. Outcomes is a list of Praxis-like sentences describing the new state of the database if the action is performed, where new sentences are either inserted or deleted from the database.

4.2.2. Logic Database

We use an exclusion logic database as it is well suited for finite state machines and aligns with the way that Praxis, the DSL of Versu, was designed. The database utilizes our version of Praxis, Praxish, to provide a simplified formulaic sentence that the database can easily understand and quickly change its state based off of new sentences being queried.

4.2.2.1. Praxish

Praxish is our DSL (Domain Specific Language) for Wyclef, trying to emulate what Praxis does for Versu. From studying Versu’s documentation on Praxis, we were able to discern a few key components for rebuilding Praxis: usage of periods and bangs (exclamation points), capitalized vs. non-capitalized words, and operators.

```
"NOT practice.tendBar.Bartender.customer.Actor!order!Beverage"
```

Figure 5: Praxish Sentence

In the example Praxish sentence (figure 5), we see these three components at play. The bangs and periods between each word of the sentence denote whether the next layer (word) can have multiple children, a period, or just one child, a bang. Capitalization of a

word indicates it is a variable and non-capitalization means it is a constant. Finally the operators are separated from the sentences with a space and can be also used with variables retrieved from the database through Praxish sentences. For example, using the variables Bartender and Actor, from the above sentence, we could then form a Praxish statement “NOT Bartender Actor” which when used in a database query would keep only the states that have the Bartender not the same as the Actor taking the action.

4.2.2.2. Database Methods

The database has a set of methods that external components can call to insert, retrieve, and delete information in the form of Praxish sentences. These methods consist of insert, remove, unify, and query. All of the methods utilize a parseSentence utility function to break down a Praxish sentence into pairs of terms and a period (non-exclusive) or a bang (exclusive). Insert and remove are self-explanatory, inserting and deleting Praxish sentences from the database. Unify takes a sentence and binds variables taken from that sentence with states that match in the database.

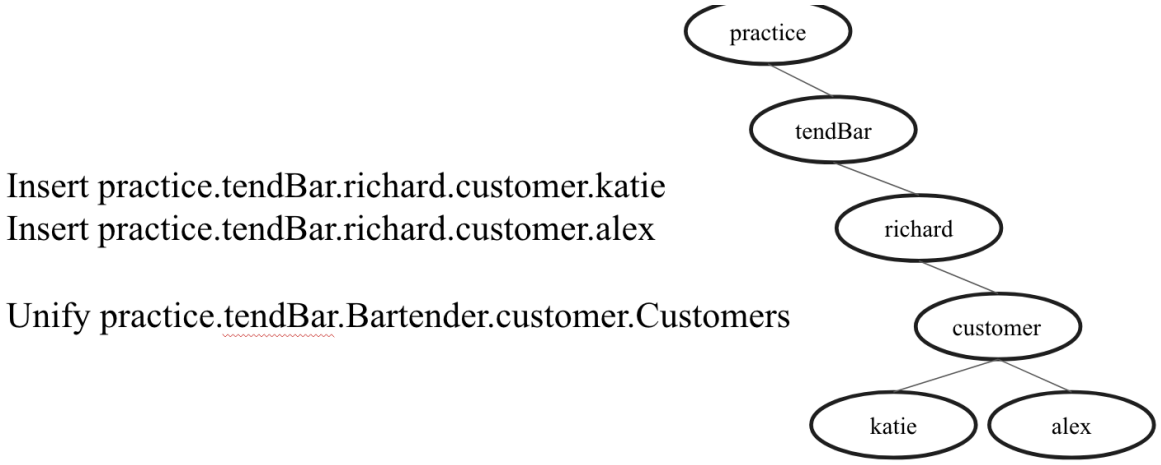


Figure 6: Example of Unify/Query

Using figure 6 as an example, two Praxish sentences are inserted into the database and then are unified with the next sentence. The result of calling the unify statement would be this list of bindings: [{ Bartender: Richard, Customers: Alex},{ Bartender: Richard, Customers: Katie}]. We can see there is a state where Richard is a bartender and he has a customer Alex and another state where Richard has a customer Katie. Query is similar to unify but it additionally has operators like NOT, EQ, and NEQ, and can also be used on a set of statements to produce a set of potential variable bindings that are consistent with *all* of these statements. This method is the core of Versu, as it is used to check which actions are plausible in the database's current state.

There are a few unimplemented methods that we plan to implement in the future, such as the conditional operator. The conditional operator can be used to conditionally perform action outcomes, allowing a single action to update the database in multiple different ways depending on the database's current state. The call method is another feature we plan to add. It calls the user's functions on the database and can also call a combination of any of the database methods as well. In addition, an increment/decrement operator will allow actions to increase and decrease numeric values when performed, enabling features akin to Dungeons and Dragons' attribute system and dice rolls. All of these methods together form the core of the Wyclef Social Simulation engine.

4.3. Connecting the Frontend and Backend

This layer of development is for utilizing Praxish to build out an example of a Social Simulation. This layer in combination with Praxish is Wyclef. The view controller contains functions to get all possible actions for a character given the current state of the database, initialize characters and practices to be used, and take an action specified by the player for the currently acting character. It also contains JavaScript for the frontend functionality. At the core of the functions above, is one or more of the database functions in conjunction with the code relating to which characters are supposed to be affected or

the database functions relate to. When the get all actions function is called, it is specifically to get all actions for the character who is currently being considered – in other words, the character whose turn it currently is to act.

5. Test Plan and Results

5.1. Test Plan

One of the most important parts of game development is playtesting. This involves sitting down with the current build of the game, and playing through it. Optimally, it involves the game's developers silently watching a user play through the game without giving any help or advice on how they should play.

Our original test plan consisted of doing just that, sitting down with a student detached from the project and observing them play with Wyclef, while taking note of pain points and places that needed to be further fleshed out. Due to time constraints, we did all playtesting internally between the team and the professor.

A playtesting session went as follows: We started up Wyclef with a bartending practice giving a world location of a bar and asked Dr. Kreminski to choose a few names for characters. After the characters have been added to the database, we started the game and let Dr. Kreminski take the role of one the character of their choosing, and then the rest of the team fulfilled the role of the remaining characters. As actions are taken and preconditions are met, the opportunity for different actions opens up and their outcomes insert new practices to the social setting.

Another variation of playtesting consisted of trying to run through every possible interaction, ensuring the state of the database afterwards was correct. This was the most common form of playtesting as it could be done by just a single developer.

5.2. Results

Through the rational reconstruction process, there were a lot of takeaways to be had. Wyclef in its current state is not yet a full reconstruction of Versu, but the progress made has proven the worth of the Versu Social Simulation Engine. In the current state Wyclef has:

- The core of the exclusion logic database completed
- A frontend to allow debugging of the engine and to test social practices
- A template for the authoring of additional social practices in JSON

The goals set out for Wyclef during the duration of the senior design project were for the most part achieved, with a few features still in the works as stated earlier. More database functionality (specifically the addition of the conditional, call, increment, and decrement operators) is the top priority for further development. Adapting Versu into a JavaScript-based engine immediately gave us the idea of using JSON to format practice definitions, which both made it easier for content authors to rapidly create new practice definitions and substantially simplified our practice definition parser in comparison to Versu's (which used a fully novel syntax for practice definitions). After working through making a few practices and testing them with the frontend, the authoring of new practices became much easier. There is still a lot of work to be done, described in 7.3 Next Steps.

6. Societal Components

While we try to create a project that is well thought out in a technological sense, we also have to take into account the positive and negative aspects of our project when it pertains to the societal components. Our project does not aim to solve any societal dilemmas yet we are not free from not having unwanted effects on not only people but also the Earth.

6.1. Ethical Concerns

Due to Wyclef being an open source reconstruction of proprietary technology, we have to be careful of who gets involved in the work. Versu is not patented, and reconstructing it from only publicly available documentation does not incur any legal issues. However, if the original creators of Versu wanted to contribute to future Wyclef development, this could cause legal issues with the company that still owns the rights to Versu.

As stated before, our project is centered around our users creating their own stories in the game. To do this, users must take actions through a character. To create a character, a user adds a preferred name for them, then the character is ‘born’ with a set of characteristics that currently are: age, height, weight, eye color, and hobbies. Since their characteristics are currently not used for their stories, they are not visible nor accessible to the user. Some concerns that come with the current way that we are handling characteristics is that there might be a lack of representation due to the limited amount of characteristics currently available. This lack of diversity might make the user feel disconnected from their character since they might feel that this character is not portraying someone they want in their story or not feel like they can insert themselves into the story. In addition, since these characteristics are not open to being edited by the user, they might get a set of qualities that they do not feel satisfied or comfortable with. It is inevitable for people to become immersed in stories and see themselves in the place of the character they are playing, but if the characteristics are something they shy away from then the story will not feel authentic. To mitigate these feelings, in the future, we wish to add a variety of characteristics that are easily recalibrated to align with the image that the user has in their mind for their character.

6.2. Sustainability and Environmental Impact

Wyclef is a project that is completely based on computers so the environmental impacts that we would have to be aware of are our part in the increasing use of computers. We have to recognize the use of resources and labor that goes into building hardware, and the resources and maintenance used to have the servers that are being used to keep our project online up 24/7. The sustainability aspect of our project is that the people using Wyclef will probably be using computers that they already have, so we are not demanding them to buy something new and does not require the usage of servers as it is run on the localhost.

6.3. Social Context

Video games are an integral part of the childhood of countless children worldwide, but with their rising popularity comes criticisms from parents and other worried adults. A criticism that has been brought up is that more kids are becoming addicted to video games which are taking away their time from making friends or focusing on their studies. Since our project aims to enrich the gameplay of games, this will motivate players to keep playing to see what new adventure awaits them depending on what action they take. While this will improve the quality of games for the average user, it will also enhance the experience of users that are addicted to playing video games. Some may say we are encouraging their addiction instead of weaning them off games but we want to drive home the importance of individual responsibility. We don’t want to make video games bad so kids won’t stay on them for too long, we want them to have a good experience while playing during the time permitted to them by their parents. We want video games to be an outlet for them to be creative and later expand that creativity outside of video games and into their everyday life. An example of this is if a child is interested in games that have cool animations, they may find a passion in drawing or they may want or if they like the storyline of a game it may inspire them to write stories of their own.

6.4. Economic

Wyclef is an open-source project posted on GitHub for anyone to see, download, and use in the development of their own projects. We made this project open for all because we are recreating a tool, Versu, that has been placed behind a paywall, so we found it imperative to make a similar tool accessible to the general public again. We don't want anyone to be limited in the tools they can use because of their financial status; therefore, our project needed to be free for anyone to use. Additionally, developing Wyclef in the open creates the possibility that other developers will be able to pick up where we left off in the implementation of intended features that we left for future work due to time constraints. Wyclef is a starting point where others can add onto and edit Wyclef to best fit their needs.

As stated before, we are recreating Versu, a tool that still exists, so some might think that our tool will take away profit from the company that it is currently under. However, the company that currently owns Versu is neither using it in their own projects nor making it available to others, and has also declined to sell the project back to the original team that worked on it; consequently, Versu is largely thought of as lost technology. Since the company that owns Versu is not currently making any money from the project, we would not be taking away their clients by releasing an open-source alternative – just giving people a chance to gain access to similar technology.

6.5. Health and Safety

Concerns related to Health and Safety are relatively minimal in the case of Wyclef, as there isn't something tangible to be touched. As it is a social simulation engine, it is a program that is used to build more fleshed out programs and is currently set up to work without need of servers so the only physical thing the user will need to operate is their computer.

6.6. Manufacturability

The manufacturability of Wyclef is also not a pertinent topic as it is software that is currently being stored in a repository. Rebuilding Wyclef is not a worry of ours or our users, but the manufacturability of Social Simulations built from Wyclef is a concern of ours. This is the reasoning behind the usage of JSON for the social practices, as it creates a general format for users to use even in other similar reconstruction of Versu that may appear in the future. The portability of JSON into a variety of languages makes the manufacturing process of a new reconstruction of Versu much simpler due to the ability to repurpose the social practice standard that we have constructed.

6.7. Usability

There are not many concerns in regards to usability. The project is open source, so access to the technology is easy, and due to the wide usage of JavaScript, the engine should be easily modifiable by those who want to expand upon it in the open source community.

7. Conclusion

7.1. Summary

Wyclef sought out to reconstruct Versu, and while in its current state it has not fully accomplished this goal, it has completed the foundation of this goal. With the exclusion logic database built, the core of the social simulation engine is complete. This enables future researchers on the project to quickly iterate on new additions to the project as well as opening the contribution to the open source community. All of the goals set out in functional requirements were accomplished during the time period of the project which was a major success.

7.2. Lessons Learned

There were a lot of lessons throughout the project. One of them was that building the foundation of the database made other parts of the project easier. This was a lesson in seeing the forest through the trees. Some tasks seemed more daunting than they really were once other parts of the project had been fleshed out. An example of this was creating a format for the JSON objects that hold the social practices. Once the practice parser had been implemented, making a template format for the social practice JSONs was quite simple. In addition, having a frontend really helped with debugging iterating through versions of the back-end since before hand they had no way of being visualized, and also allowed the usage of the browser console.

7.3. Next Steps

We plan on furthering the ease of authoring and implementing the world and character parsers. In particular, we plan to recreate a version of Versu's higher-level Prompter language (Nelson, 2014), called Promptish, to allow the authoring of practice definitions and other Wyclef content in a natural-language syntax, rather than requiring content authors to write the lower-level Praxish domain specific language directly. We will also implement equivalents to Versu's world and character files, including parsers for each; these files will be used by content authors to further flesh out the database, for example enabling characters to have outcomes they desire and make comments on other characters' actions based upon their biases. This will involve modifying the action executor to ensure that characters prefer performing actions that further their goals. Once world files are implemented,characters will be able to be assigned a location and restricted to only interacting with others in that location, except for via actions that cross location boundaries (such as a phone call or a television broadcast). Finally, additional functionality for Praxish and the exclusion logic database will be added through a call

method and a conditional method. See 4.2.2.2 Database Methods for more details on these additional methods.

8. Bibliography

McCoy, Joshua, et al. “Social Story Worlds With Comme il Faut.” *IEEE Transactions on Computational Intelligence and AI in Games, Vol. 6 ...*, June 2014, cs.uky.edu/~sgware/reading/papers/mccoy2014cif.pdf.

Nelson, Graham. *Prompter: A Domain-Specific Language for Versu*, May 2014, versublog.files.wordpress.com/2014/05/graham_versu.pdf.

Short, Emily. “Versu: Content Structure.” Emily Short’s Interactive Storytelling, 22 Feb. 2013, emshort.blog/2013/02/22/versu-content-structure/.

Short, Emily, and Richard Prideaux Evans. The AI Architecture of Versu, May 2014, versublog.files.wordpress.com/2014/05/versu.pdf.

Stack Exchange, “Stack Overflow Developer Survey 2020.” *Stack Overflow*, 2020, insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers.

Thompson, Tommy. “The Story of Facade: The AI-Powered Interactive Drama.” Game Developer, 23 Apr. 2020, www.gamedeveloper.com/design/the-story-of-facade-the-ai-powered-interactive-drama.