# Analysis of Social Gameplay Macros in the Foldit Cookbook

Seth Cooper[1], Firas Khatib[2], Ilya Makedon[1], Hao Lu[1], Janos Barbero[1], David Baker[2], James Fogarty[1], Zoran Popović[1], and Foldit players[3]

[1] Center for Game Science
Department of Computer Science & Engineering
University of Washington
{scooper,makedon,hlv,jbarbero,jfogarty,zoran}@cs.washington.edu

[2] Department of Biochemistry
University of Washington
{firas,dabaker}@u.washington.edu

[3] Worldwide

## ABSTRACT

As games grow in complexity, gameplay needs to provide players with powerful means of managing this complexity. One approach is to give automation tools to players. In this paper, we analyze an in-game automation tool, the *Foldit cookbook*, for the scientific discovery game Foldit. The cookbook allows players to write *recipes* that can automate their strategies. Through analysis of cookbook usage, we observe that players take advantage of social mechanisms in the game to share, run, and modify recipes. Further, players take advantage of both a simplified visual programming interface and a text-based scripting interface for creating recipes. This indicates that there is potential for using automation tools to disseminate expert knowledge, and that it is useful to provide support for multiple authoring styles, especially for games where the final game goal is unbounded or hard to attain.

## Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: General – Games

## 1. INTRODUCTION

As games grow in complexity, gameplay needs to provide players with powerful means of managing this complexity. One approach is to give tools for automation to players. This gives players the opportunity for customization and allows them to approach the game at a higher level. World of Warcraft [20] allows macro commands and also a rich scripting language for creating addons. Second Life [15] has a scripting language that allows users to define the behavior of objects. Final Fantasy XII [7] introduced the gambit system, which allowed players to automate many of the details of battle.

The *Foldit cookbook* was developed to aid players in systematizing their strategies and integrating them into the social environment of a scientific discovery game. In this paper we show the potential of automation in a social context for propagating the expert skills of top Foldit players and increasing the overall collective problem solving skills of the predominantly non-scientist Foldit player population. In addition, we discuss our experience deploying the cookbook to the Foldit community at large, present analyses of two data sets gathered from players who used the cookbook, and conjecture directions for future improvement of in-game automation in scientific discovery games.



FDG'11, June 29-July 1, Bordeaux, France


## 2. RELATED WORK

Programming generally requires sophisticated skills and training. Extensive prior work has presented new programming languages to make the task easier and require less training. Among them, visual programming languages [2] have been popular, particularly in educational contexts. They often take the form of assembling blocks of code, such as in Scratch [13] and StarLogo TNG [18]. Alice [4] is a visual programming language designed to teach students programming through storytelling. Kodu [11, 12] is a visual programming language designed for young children to design and program video games. The Foldit cookbook design faces the same challenges in supporting user programming and uses similar visual programming language techniques. The recent language Toque [16] is particularly interesting for its use of cooking metaphors, which the Foldit cookbook also employs.

Analysis of data from online games is an expanding field, giving insights into the dynamics of online games and the communities around them. Recently, Lewis [9] gathered a large data set on World of Warcraft by crawling web pages. This data was used to answer questions about the game world, such as which classes level faster or die more often. Williams [19] was given access to a large amount of data on EverQuest 2, which was used to discover the profiles of online gamers. In this paper, we perform data analysis gathered from Foldit players using the cookbook.

How users share their software customization and collaborate in problem solving has also been studied in other domains. Mackay [10] studied patterns of sharing customizable software through a study of users at two research sites working with two different kinds of customizable software. Three groups of sharers were found: the experts who created customization from scratch, the translators who created simplified and more task-specific versions of the ex-


**Figure 1: A screenshot of the cookbook editor for a GUI recipe. The visual blocks that make up the recipe take up the main section of the window. Buttons for saving, loading, sharing, and other options are along the bottom.**

pert customizations, and the rest of the organization who adopted customizations from the translators. Berlin and Jeffries [1] studied interactions between experts and apprentices in learning and collaborative problem solving. One primary strategy found in their work was "copy and experiment": the apprentices find something similar to what they want, copy it, and then try to customize it to their specific needs. In our Foldit cookbook, the sharing activities occur in a much larger scale and in a much more distributed setting. Our analyses discover some of these same behaviors in scientific discovery games and also make new observations about group-based sharing and ratings.

# 3. OVERVIEW

## 3.1 Foldit

Foldit is a multiplayer online game in which players compete to find well-folded protein structures. See [6] for a more detailed description of Foldit's design. In Foldit, protein structures are posted online as puzzles, accessible for fixed periods of time, usually one week, during which players compete to achieve the highest scoring solution structures. Players interact with the protein in a variety of ways: they can pull on it directly, place *bands* to pull indirectly, and *freeze* pieces to prevent them from moving. They can also launch optimizations, called *shake* and *wiggle*, that will computationally improve their protein. The score of the player's solution is updated in realtime. Foldit is organized as a competition, and a leaderboard of all other players and their scores is displayed. Players can form groups, which can work together and share their solutions. Groups compete on a separate group leaderboard.

Foldit uses biochemical modeling, evaluation, and optimization routines from the Rosetta package [14] for scientific accuracy. It has been shown that Foldit and its players are capable of solving hard structural biochemistry problems, especially those that are known to be problematic for state-of-the-art computational methods [5].

## 3.2 Cookbook

Soon after Foldit's release, the players created a wiki [8]. One section of the wiki was devoted to player strategies; players began requesting the addition of automation tools

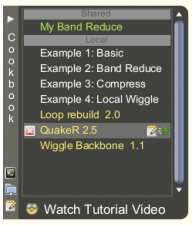


**Figure 2: The cookbook. Recipes are divided into sections based on whether they have been shared. When the mouse hovers over a recipe name, buttons for running, editing, and deleting appear.**

so that they could more easily carry out their strategies. It was also our intention to infer optimal strategies from the Foldit players and use them to improve fully automatic approaches. Rather than performing machine learning on gameplay traces of Foldit players, we decided that the players themselves would likely be much better at systematic abstraction of their strategies. For this reason we added the *cookbook* to Foldit. The cookbook allowed players to write, share, and run *recipes*, which were automated version of their strategies.

The initial interface for writing recipes was a simple block-based visual programming interface, illustrated in Figure 1. Using this interface, a recipe is created by adding *steps* from a pulldown menu. Steps include game moves such as shaking, wiggling, adding bands, freezing, or restoring saved solutions. Steps can be deleted or reordered as desired. Each step may have some number of *ingredients*, parameters that need to be specified. It is worth noting that this interface does not have support for conditionals or loops. Recipes created with this interface are called *GUI recipes*.

After the addition of the cookbook with GUI recipes, players additionally requested the expressive power of a full scripting language. We therefore added an alternative text-based interface, using the Lua scripting language [17]. The base Lua language was augmented with custom functions to execute game moves and query game state. The Lua interface has many more available moves and the ability to control the flow of the recipe, allowing much more expressive recipes to be written. Recipes created with this interface are called *script recipes*.

Players are able to upload their recipes to the game servers to share with other players. They can share a recipe globally with all players or only with players in their group. The Foldit website has an interface for players to search for shared recipes. Each recipe has a webpage with information about the recipe, including its title, description, author, and comments. Each recipe additionally has a five-star rating. Players can rate a recipe on its webpage and are also reminded to rate, if they have not yet, when running a recipe in-game. If a recipe has been created by modifying a previous recipe, the recipe it is derived from (its parent) is listed on the webpage; any derived recipes (its children) are also listed. There is a button on the webpage that will add the recipe to the player's cookbook. The player who shared a recipe can make new revisions of it after it has been uploaded. These revisions will be considered the same recipe, and only the newest revision can be downloaded.

| | |
|---|---|
| Puzzles | 56 |
| Recipes | 5488 |
| Recipes run >= 10 times | 1139 |
| Recipes run >= 100 times | 233 |
| Recipes run >= 1000 times | 26 |
| Recipes run >= 10000 times | 1 |
| Total times recipes were run | 158682 |
| Players | 3590 |
| Players who ran a recipe | 771 |
| Players who wrote a recipe | 568 |
| Recipes written during CASP9 | 5202 |

**Table 1: Overview of data from the CASP9 data set.**

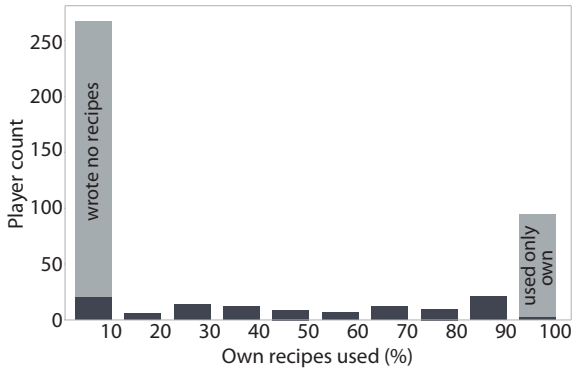


**Figure 3: Histogram of recipe users by the percentages of recipes used that they wrote themselves for recipes in the CASP9 data set. The lighter bar on the left shows players who used only other players' recipes; the presence of this bar indicates that the expertise of players who write recipes can be used by players who do not write recipes. The lighter bar on the right shows players who used only their own recipes.**

While playing Foldit, a list of the recipes in a player's cookbook are given in a collapsible sidebar in the game, shown in Figure 2. As an introduction for players, the cookbook comes with four example GUI recipes and a button that links to a video introducing the cookbook. There is also documentation available for the steps and function calls, as well as text bubbles that appear explaining each step the first time it is used. The player can easily run, edit, or delete recipes from the cookbook. While running, the cookbook is replaced by text that gives the progress of the current recipe. Individual steps within the recipe can be canceled, or the entire recipe can be canceled.

# 4. CASP9 ANALYSIS

To understand the social aspects of the cookbook, we analyzed data gathered from the puzzles run for CASP9 [3]. CASP is a biannual event in which teams compete to most accurately predict protein structures. Foldit ran a series of CASP9 related structure prediction puzzles over a period of several months, from May 6 to August 19 in 2010. Basic information about the puzzles, players, and recipes used can be found in Table 1. In order to ignore recipes that were used a very small number of times, our CASP9 analyses consider only the recipes that were run at least 10 times.

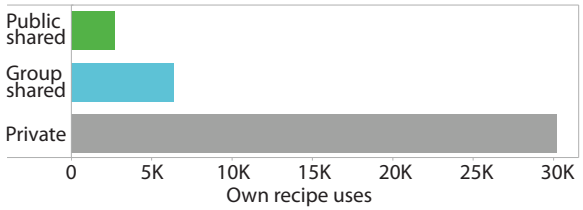


**Figure 4: Sharing status of recipes run by players who run only their own recipes in the CASP9 data set. Most of the recipe uses are of private recipes, indicating that there is room to improve incentives for sharing recipes.**

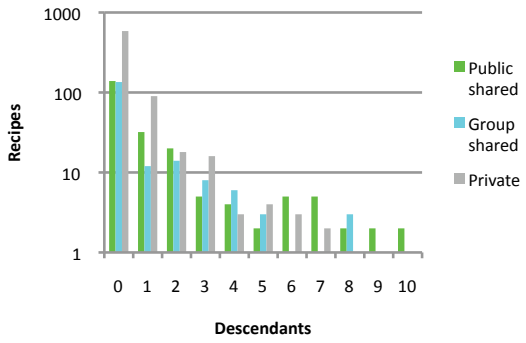


**Figure 5: A histogram of the number of recipes from the CASP9 data set by number of descendants. There were 9 public shared, 3 group shared, and 4 private recipes with more than 10 descendants. While most recipes have no descendants, some recipes led to a variety of modifications and customizations, creating many descendants.**

## 4.1 Recipe Sharing

As Foldit players are able to share recipes with other players, we were interested in observing the patterns of use of recipes written by other players. This would allow us to determine if, and how, players were sharing recipes and using shared recipes.

In order to understand how players use shared recipes, we analyzed the patterns of use of recipes written by players themselves and those written by other players. We considered all players that either used a recipe during CASP9 or wrote a recipe that was used. For each of these players we computed the following statistics: total number of recipes used, total number of recipes written, number of times the players used recipes written by others, and number of times other players used recipes written by the player.

This led us to identify two distinct classes of recipe users: those who mostly used recipes that they wrote, and those who mostly used recipes written by others. We calculated what percentage of total recipe uses by each player were recipes written by that player. Figure 3 shows the histogram of the usage of a player's own recipes by 10% bins.

The large number of players in the leftmost bin correspond to players who primarily used recipes written by other players. The gray area identifies the number of pure consumers,

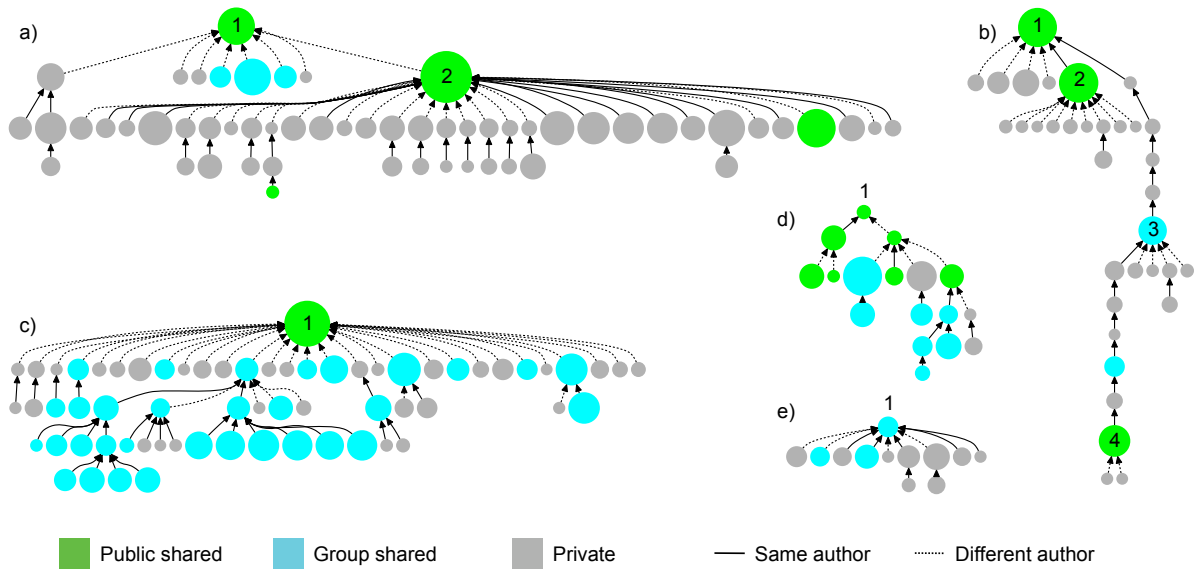


**Figure 6: Example hierarchy of recipe descendants from the CASP9 data set. A child is created when a player edits a previously existing recipe. Size is logarithm of number of uses of a recipe. The ability to modify recipes shared by other players led to a variety of interesting modification patterns.**

players who wrote no recipes at all. Combined, these 246 players used recipes over 36,000 times. It is worth noting that 13 of the pure consumers account for more than half of these uses.

The rightmost bin corresponds to players who mostly used recipes they wrote themselves. Looking more closely at players who ran only their own recipes, Figure 4 shows the number of times that recipes were used by their authors, broken down by their sharing status. Clearly, these players tend to have an arsenal of recipes that they keep to themselves.

This data shows that the recipe sharing system is being used by the players and that there are distinct patterns of use among players. The high number of pure consumers indicates that recipes are being used by players who do not write them, and thus have the potential to facilitate the exchange of expertise from players who are writing recipes. However, there are a number of players who are running recipes that they have not shared. More could be done to more to encourage these players to share their recipes, potentially by enabling players to earn points for others using their recipes.

## 4.2 Inheritance Relationships

Given that players were using and sharing recipes, we wanted to observe if players were modifying those recipes and what patterns of inheritance they produced. When a player shares a recipe, any player can then modify that recipe and create a new recipe that becomes a child of the original. Players can also modify their own private recipes to produce children. This allows for different variations of a similar concept to be produced, and it provides an opportunity for others to improve on a player's creation.

To observe patterns of how players were modifying other recipes, we looked at the number of descendants for each recipe. A descendant is any recipe that can be traced back to through its parents an original recipe. It is worth noting that it is possible for someone to take a shared recipe, rewrite it from scratch (with or without modifications), and call it their own. This would not show up as a descendant in our data set. Most recipes in the data set had no descendants, but there were over 250 recipes with one or more of them and 16 recipes with over ten descendants (Figure 5).

To investigate further, we looked at the inheritance hierarchies for the recipes with the most descendants. Several interesting patterns are shown in Figure 6. Panel (a) in Figure 6 is an example of the most-used recipe during CASP9, *Blue Fuse v1.1* (a, 2), which was derived from another public recipe, *Acid Tweeker v0.5* (a, 1). *Blue Fuse v1.1* has many descendants, but almost half of them are from the same author of *Blue Fuse v1.1* (shown as solid lines). The same is true for the *Loop Rebuild* family of recipes shown in panel (b). *Loop Rebuild 1.0* (b, 1) and *Loop Rebuild 1.1* (b, 2) were heavily used recipes during CASP9, with many other players creating different versions of both (shown as dotted arrows). The author of *Loop Rebuild 1.0* produced many descendants of it, sharing two of them with his group along the way (shown in cyan, such as *Loop Rebuild 1.9* (b, 3)) before finally sharing *Loop Rebuild 2.0* (b, 4) with the entire Foldit community. The hierarchy for *Blue Fuse v1.1* is broad, showing experimentation from a common starting point, while the hierarchy for *Loop Rebuild 1.0* is deep, showing iteration and refinement.

Some recipe writers publicly share their creation and do not create any modifications afterward. Panel (c) in Figure 6 shows an example of a popular recipe with many direct children, none of which were written by the original author. *Quake* (c, 1), which was the second most-used recipe during CASP9, has many different descendants, yet none of these modifications were made public. Many of these *Quake* descendants were shared within groups, but not with the rest of the Foldit community. The recipe *simple wiggle by ones* (d, 1) has many publicly shared descendants that ended up be-

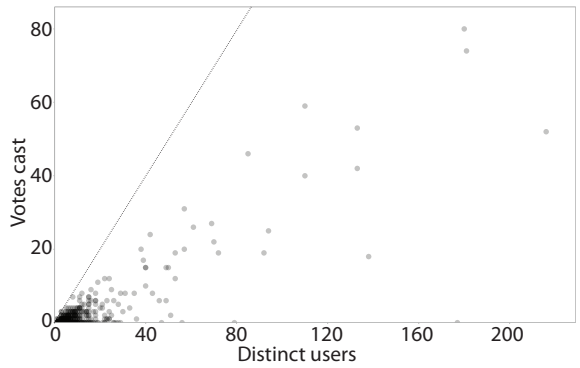


**Figure 7: Number of distinct users of a recipe and number of votes they cast for recipes in the CASP9 data set. The line shows where points would be if every user rated. The average number of votes cast per user of a recipe was 0.25. Many recipe users do not rate recipes that they run.**

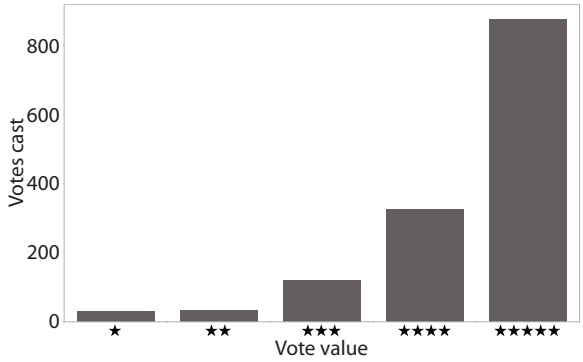


**Figure 8: Distribution of votes by rating for recipes in the CASP9 data set. Most players who cast votes gave 5 stars. Players did not take advantage of the full range of ratings available, indicating a simpler rating system could have been used.**

ing used even more than the original parent during CASP9. The recipe *Lua EAR 3-5 LSR* (e, 1) was shared with the author's group and led to the creation of more group shared recipes.

Through this analysis it appears that players are taking advantage of the ability to modify and adapt the recipes of other players. Publicly shared recipes have the potential to create many more descendants with wide inheritance hierarchies, but keeping recipes private also provided value to players and allowed more localized refinement.

## 4.3 Ratings

The Foldit cookbook gives players the option of rating the recipes they use. Players can rate on the webpage for a recipe or in game via a popup. We wanted to determine if players were rating the recipes they used and what kind of ratings they were giving.

The number of users of a recipe who rated the recipe is shown in Figure 7. We found that it was difficult to get Foldit players to rate each other's recipes. While recipes with more unique users have more ratings, many users did not rate the recipes they used. Even with a pop-up to solicit ratings after using a new recipe several times, many users still did not rate the recipes.

We used a 5 star rating system, with 5 being the highest rating. However, we found those who did rate recipes mostly rated them with 5 stars (Figure 8). It is possible that players simply stop using recipes that they do not like, without bothering to rank them. Another possible explanation is that players do not want to negatively rate other players' creations, for fear that it might cause an author to no longer share their future recipes.

To improve feedback from ratings, we could require players to rate a recipe after using it many times before they are allowed to use it again or delete it. Perhaps a simple question, such as "Do you like this recipe?" with "Yes" and "No" buttons, or a "Like" feature might have yielded more ratings. A simple choice might have made it easier for players to decide how to rate. It might also be possible to have the option to vote always available in game while a recipe is running, rather than only popping up occasionally.

# 5. SCRIPT RECIPE ADOPTION ANALYSIS

We analyzed data around the time of the introduction of script recipes to discover how the introduction of script recipes impacted the existing GUI recipes. Script recipes give players the option to write recipes based on the Lua scripting language in addition to the original GUI recipes, and were released several months later. The cookbook editor defaults to creating a GUI recipe, and there are two "New" buttons (one for each type of recipe). This data was gathered from July 2009 to April 2010.

Figure 9 shows numbers for recipes written and used during this time. We can see that script recipes quickly became popular to use. However, it took many weeks before their popularity overtook the popularity of GUI recipes. In addition, it appears the use of GUI recipes remained popular and did not decline. Further, both script and GUI recipes continued to be authored at this point.

This indicates that GUI recipes continue to be useful even though the more powerful script recipes have been introduced. Therefore, it can be useful to provide both simpler visual methods and more advanced automation capabilities in scientific discovery games.

# 6. CONCLUSION

Foldit tackles a difficult scientific problem, requiring players to master many different skills in order to successfully fold protein structures. Players have come up with various strategies to perform well in Foldit. Allowing Foldit players to create and run recipes was a method of codifying these strategies, and including the ability to share recipes has been instrumental in disseminating player strategies. Facilitating the sharing and modification of other's recipes is a method of distributing expertise. Players are easily able to run shared recipes without the prerequisite of being able to write them and may come up with different fruitful ways of using them that the authors may not have considered. Conversely, a player may not be the best at folding proteins

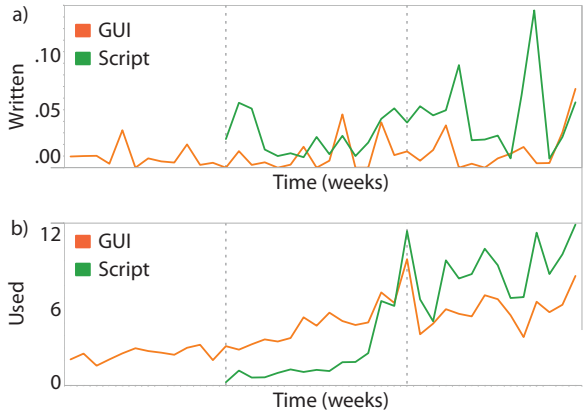


**Figure 9: Comparison of GUI and script recipes around the time of the introduction of scripting, binned by week and normalized by number of unique competing players. Shown are a) number of recipes written and b) number of recipes run. The first vertical bar indicates when script recipes were introduced. The second indicates a refinement of the web interface for recipes. Even after the introduction of script recipes, GUI recipes continued to be written and used, indicating that both interfaces were useful.**

but is a valuable asset to other Foldit players if they produce useful recipes.

We have found that although most recipes have no descendants, there are still many shared recipes that have been modified by other players, often leading to more descendants. One way to increase these numbers would be to incentivize recipe sharing. This could be done by allowing players to earn points for others using their recipes, but this would require hiding the details of another players' recipe or they could easily be copied.

Despite the use of shared recipes, we found it difficult to get players to rate recipes they used. If players did rate, they was typically positive ratings. We would like to experiment with different incentives to encourage players to rate recipes they use, and simplify the rating system to reflect these usage patterns.

We found that both the visual GUI recipes and text-based script recipes were written and used despite vast differences in interface and expressiveness. This indicates that support for multiple authoring styles can be beneficial. We would like to further examine why players choose one interface over another and what kinds of recipes or styles of writing are best supported by each interface.

## 7. ACKNOWLEDGMENTS

We would like to acknowledge the members of the Foldit team for their help designing and developing the game and all the Foldit players who have made this work possible. This work was supported by the Center for Game Science, DARPA grant N00173-08-1-G025, the DARPA PDP program, NSF grants IIS0811902 and IIS0812590, the Howard Hughes Medical Institute (D.B.), and Microsoft. This material is based upon work supported by the National Science Foundation under Grant No. 0906026.

## 8. REFERENCES

[1] L. M. Berlin and R. Jeffries. Consultants and apprentices: observations about learning and collaborative problem solving. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work - CSCW '92*, pages 130–137, New York, New York, USA, Dec. 1992. ACM Press.

[2] M. Boshernitsan and M. Downes. Visual Programming Languages: A Survey, 2010.

[3] CASP. http://predictioncenter.org/.

[4] M. Conway, S. Audia, T. Burnette, D. Cosgrove, and K. Christiansen. Alice: lessons learned from building a 3D system for novices. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, pages 486–493, New York, New York, USA, Apr. 2000. ACM Press.

[5] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, and F. Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010.

[6] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, and Z. Popović. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 40–47, New York, NY, USA, 2010. ACM.

[7] Final Fantasy XII. http://www.finalfantasyxii.com/.

[8] Foldit Wiki. http://foldit.wikia.com/.

[9] C. Lewis and N. Wardrip-Fruin. Mining game statistics from web services: a world of warcraft armory case study. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 100–107, New York, NY, USA, 2010. ACM.

[10] W. E. Mackay. *Patterns of sharing customizable software*. CSCW '90. ACM Press, New York, New York, USA, 1990.

[11] M. MacLaurin. Kodu: end-user programming and design for games. In *Proceedings of the 4th International Conference on Foundations of Digital Games - FDG '09*, page xviii, New York, New York, USA, Apr. 2009. ACM Press.

[12] M. B. MacLaurin. The design of kodu: a tiny visual programming language for children on the xbox 360. *SIGPLAN Not.*, 46:241–246, January 2011.

[13] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The scratch programming language and environment. *Trans. Comput. Educ.*, 10:16:1–16:15, November 2010.

[14] C. A. Rohl, C. E. Strauss, K. M. Misura, and D. Baker. Protein structure prediction using Rosetta. *Methods Enzymol*, 383:66–93, 2004.

[15] Second Life. http://secondlife.com/.

[16] S. Tarkan, V. Sazawal, A. Druin, E. Golub, E. M. Bonsignore, G. Walsh, and Z. Atrash. Toque: designing a cooking-based programming language for and with children. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2417–2426, New York, NY, USA, 2010. ACM.

[17] The Programming Language Lua. http://www.lua.org/.

[18] K. Wang, C. McCaffrey, D. Wendel, and E. Klopfer. 3d game design with programming blocks in starlogo tng. In *Proceedings of the 7th international conference on Learning sciences*, ICLS '06, pages 1008–1009. International Society of the Learning Sciences, 2006.

[19] D. Williams, N. Yee, and S. E. Caplan. Who plays, how much, and why? debunking the stereotypical gamer profile. *Journal of Computer-Mediated Communication*, 13(4):993–1018, 2008.

[20] World of Warcraft. http://us.battle.net/wow/en/.