PEPTIBLOCKS

Final Report

April 9, 2015

Christian Euler, Ding zhu, Zhong Yan

ECE1778 (2472 Words)

Introduction

PeptiBlocks is a puzzle game which users play to help researchers solve protein folding problems. The game itself is played in a three-dimensional environment with regularly-shaped blocks which represent structural elements of proteins whose folds have not been solved. When users solve the puzzle, they are also solving the structure of the protein in question; averages of the solutions provided by many users could then be used to give researchers an idea of the structure of the protein. Structural information about proteins is invaluable to researchers in a variety of fields, as protein structure is inherently linked to function[1]. Therefore a complete understanding of a protein's function requires precise knowledge of its structure.

Proteins have a range of functions with medical, industrial, and fundamental implications which make them interesting[2]. A powerful example is the HIV protease protein, whose function was known for many years, but whose structure remained a mystery since its discovery. When the citizen science game FoldIt allowed users to solve the structure, they did so in just three weeks[3]. The protein is now a drug target for treating HIV infection; an understanding of its structure was the key piece which prevented this from happening. PeptiBlocks aims to follow FoldIt with a simpler, mobile-based game that a broad audience would enjoy playing without having to understand the intricate chemistry involved in protein structure determination.

Apper Context

This purpose is particularly relevant to the research of our apper, Christian Euler. His research is in the fields of Metabolic and Protein Engineering; it involves engineering the rapid dynamics of proteins to control metabolism in microorganisms for the production of valuable chemicals such as biofuels. Engineering regulation of metabolism is not a strategy typically employed by Metabolic Engineers, as the tools to do so are relatively new[4]. This has led to the productivity-yield tradeoff: when you ask a microorganism to produce something it doesn't normally produce, it doesn't grow well. This tradeoff exists because engineered microorganisms cannot dynamically control the partitioning of their limited resources, so they use food that should be for growth to make a target chemical instead[5].

Our apper's hypothesis is that proteins could be engineered to allow microorganisms to do exactly this. However, a major challenge associated with this function-based engineering of proteins is the fact that predicting the protein structure required for a target function is a very complex problem. PeptiBlocks would help to solve this problem by providing users with engineered enzymes as puzzles for them to solve. In solving the structures of these engineered proteins, users would be providing Christian and other Protein Engineers with a deeper understanding of the possible structures of their designs. This would greatly reduce the number of proteins that would have to be produced, isolated, and tested in practice; it would essentially reduce the space that protein engineers would have to search in order to find a protein with the desired function. This stage of engineering is the most time-consuming and costly, so PeptiBlocks could greatly reduce the resources required to find a protein solution to a particular problem.

More broadly, PeptiBlocks could help to solve fundamental challenges in the field of Protein Structure and Function. Though protein-folding algorithms exist, they require massive computational power to solve even simple protein structures, because these problems are combinatorial in nature and involve millions of simulated interactions. The game FoldIt was created when users noticed that these algorithms often made nonsensical moves on the path to a solution. The human brain is much better equipped to understand three dimensional interactions than current computer algorithms, so researchers wanted to make use of it in an accessible way[6]. However, FoldIt is anything but accessible for the average person—it requires a fairly extensive understanding of how protein structure works and provides users with a limited amount of information on the topic. The power of FoldIt is in the number of users who play it, but this number is limited by its accessibility[6]. Thus, PeptiBlocks was built to be a simpler game which anyone with a smartphone could play. Since no understanding of protein biology is required to play it, PeptiBlocks could increase the throughput of crowd-sourced protein structures relative to FoldIt and therefore could allow researchers in the field to solve more protein structure problems faster without the use of computers.

Overall Design

The block diagram for PeptiBlocks is provided in Figure 1.

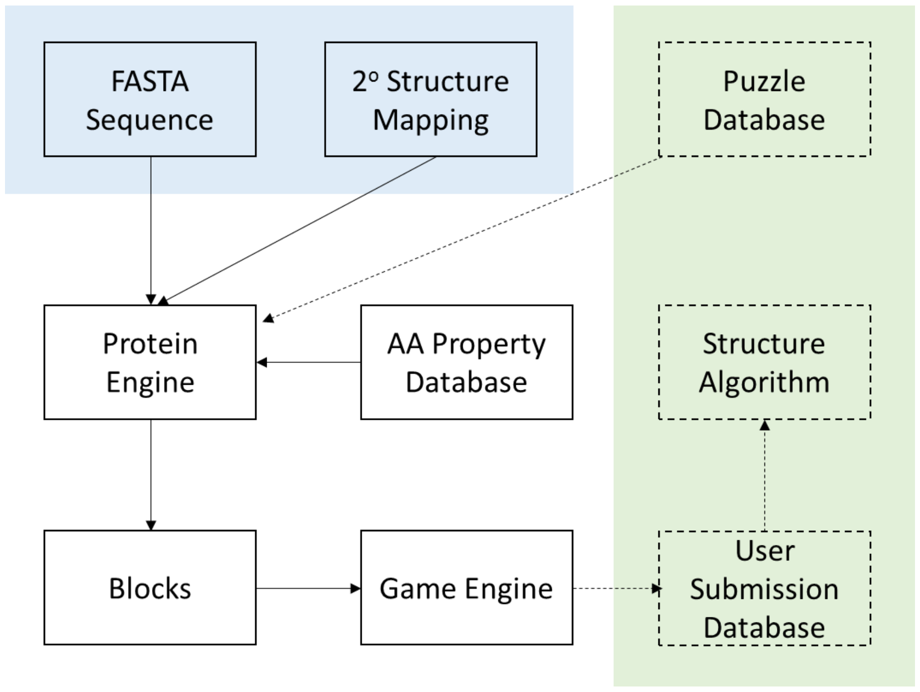


Figure 1. Block diagram representation of PeptiBlocks. Individual blocks represent pieces of the app and arrows show how information flows to and from these pieces. Blocks in a blue background are Internet-based, green blocks are associated with the app's server. Blocks with dashed lines have not yet been completed (see **Future Work**).

The core elements of PeptiBlocks are the protein and game engine blocks. In the current iteration of PeptiBlocks, the protein engine pulls a protein sequence from an online protein database (PDB.org) in the form of a FASTA file. This is simply a string of characters where each character represents one of 20 possible amino acids. The engine also pulls the secondary structure information from the PDB entry on the protein in question. It feeds this information into our algorithm to convert the protein into a series of blocks which are colour-coded according to the properties of the amino acids in the FASTA file. The colour-coding information is stored in a small database on the app (Amino Acid Property Database block). The protein engine communicates with the property database to get this information.

Information about the blocks is then passed to the game engine, which renders them in 3D by mapping amino acid properties to the blocks and then presents them to the user through its UI. The game engine controls game dynamics and integrates information provided by the user through the UI to move the blocks within its environment. The game engine also uses scoring functions to determine and present the user’s score once blocks have been placed. To date, these are the completed elements of PeptiBlocks.

Future developments involve the addition of a server to the app. This server would replace the PDB elements of PeptiBlocks, as this database only stores information on *solved* protein structures. Our server would store known amino acid sequences and calculated secondary structure information about proteins whose structures are *not* yet known (Puzzle Database block). Additionally, when users submit a puzzle, their structure would be stored on our server along with its score and user information. Solved puzzles would be averaged and converted back to 3D structures from their block forms on a server-hosted block (Structure Algorithm). See **Future Work** for additional discussion on planned changes/additions to PeptiBlocks.

Statement of Functionality

The core functionality pieces of PeptiBlocks all worked according to our current conception of the game. The major learning piece—and the riskiest part—of the project was the conversion of protein structures to blocks and implementation of 3D gameplay with those blocks. We found and were able to work with an API which allowed us to do both of these tasks well. We were additionally successful in implementing a score function which calculates the scored based on interactions between blocks in space and presents this score to users as they place blocks. The score function is capable of distinguishing between different configurations of the same sequence of blocks, which is essential to the use of PeptiBlocks as a research tool. Figure 2 demonstrates how PeptiBlocks effectively renders blocks, limits their interactions with collision-detection and distinguishes between good and bad arrangements with a score function.

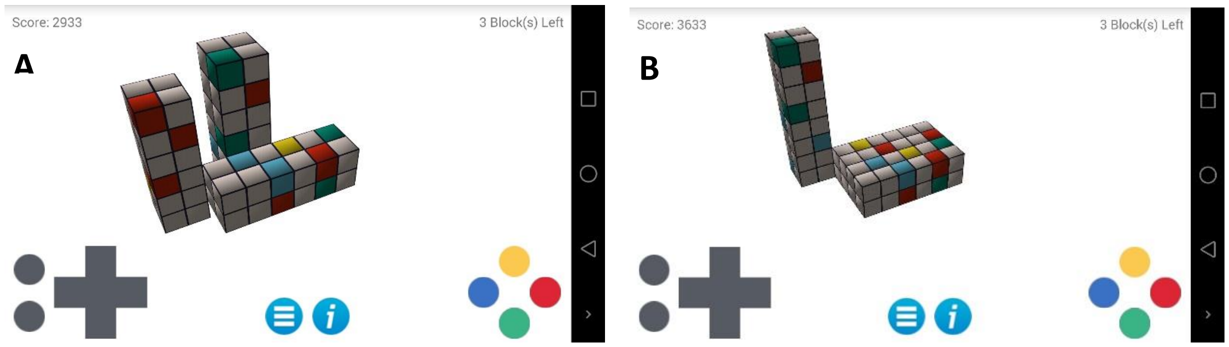
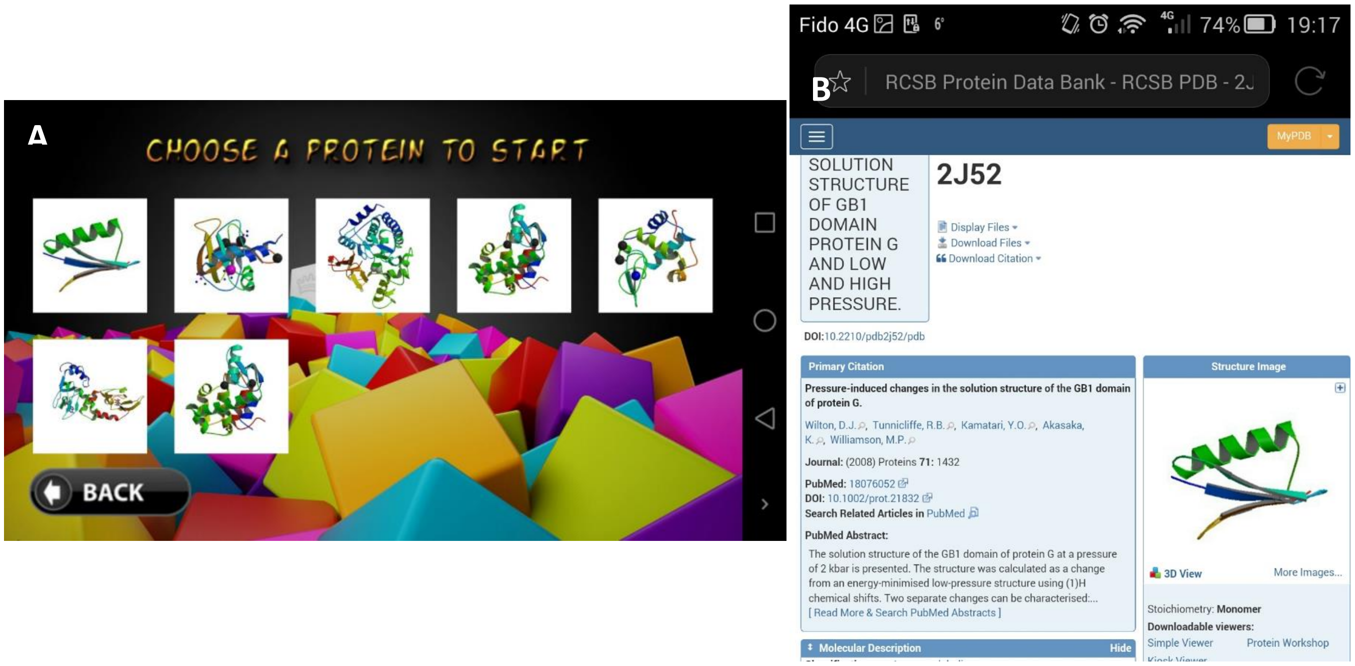


Figure 2. (A) Rendering of blocks in 3D space and (B) their interaction, controlled by users through the UI buttons and limited through collision-detection. The arrangement in (A) is less desirable than that in (B) because the first arrangement doesn't minimize space well. This is reflected in the relatively high score of (B) and the lower score of (A).

Figure 2 also demonstrates the UI we implemented, which allows users to control blocks in 3D space to orient them in such a way as to maximize score based on the game rules. These rules reflect physical interactions relevant to protein structure. Once users have placed all of the blocks in a puzzle, they can submit their solution so that it can be converted into a final protein structure.

The level selection screen, though minor, also worked well. On this screen PeptiBlocks allows users to choose from several levels, represented as 3D structure images of the proteins. Users can long tap on these images to get more information about the protein in question. This was not a planned feature, but while we were developing the app we thought it might be useful for those users who are more interested in the potential outcomes of their puzzle-solving.



SOLUTION STRUCTURE OF GB1 DOMAIN PROTEIN G AND LOW AND HIGH PRESSURE.

2J52

DOI:10.2210/pdb2j52/pdb

Primary Citation

Pressure-induced changes in the solution structure of the GB1 domain of protein G.

Wilton, D.J., Tunnicliffe, R.B., Kamatari, Y.O., Akasaka, K., Williamson, M.P.

Journal: (2008) Proteins **71:** 1432

PubMed: 18076052
DOI: 10.1002/prot.21832
Search Related Articles in PubMed

PubMed Abstract:

The solution structure of the GB1 domain of protein G at a pressure of 2 kbar is presented. The structure was calculated as a change from an energy-minimised low-pressure structure using (1)H chemical shifts. Two separate changes can be characterised:...
[Read More & Search PubMed Abstracts]

Stoichiometry: **Monomer**
Downloadable viewers:
Simple Viewer Protein Workshop

Figure 3. Level selection in PeptiBlocks. (A) Users are presented with several level options, represented by images pulled from PDB. (B) More engaged users can long tap on any of the images to get more information about the protein in question.

We could not effectively implement a few features of the app, including realtime scoring and a next block preview. We found that developing and implementing a scoring function (and collision detection) required an algorithm that could detect the location of each face of each block rapidly, as users move them. We used hashmaps to do this. The coordinates describing each block are recorded as keys in the hashmap so that they can be accessed rapidly. However, this development took some time and as a result we were not able to test it for real-time scoring. We believe that it would be possible to have real-time scoring, given that that hashmap strategy provides real-time collision detection.

A next block preview would require rendering a smaller 3D image in a larger 3D environment. Based on how steep the learning curve with 3D animation is, we decided to lower the priority of this functionality in order to focus on implementing core pieces of the app first.

Learning

We took three major learning pieces from this project. These fall broadly under communication, skills, and gamification. The results of our strategy to tackle these challenges also gave us an indication of how to better approach future projects.

Communication

A major challenge we faced as a group was a lack of knowledge about protein structure and function among our programmers. It became clear later on in the project that the root of this knowledge gap lay in the way the apper was communicating ideas. He taught the programmers about proteins in a very traditional way and in doing so took his own education in the basics of this topic for granted. Our programmers lacked this basic knowledge and didn't have time to learn it, so none of the concepts took hold. The breakthrough moment occurred when our apper used Lego to physically represent his ideas—this tactic immediately clarified the concepts for the programmers. They were able to implement the block rendering of proteins immediately following this level of communication. The lesson here is that individual communication styles are functions of previous learning and background and they must be accommodated in order for cross-discipline communication to be effective.

Skills

Our group began this project with no knowledge of 3D animation. The learning curve associated with using it was therefore necessarily steep. With no skills or knowledge with which to inform our choice, we selected the first 3D engine that we could work with intuitively. As expected, this turned out to be a sub-optimal choice; the engine we used was relatively primitive and a more mature engine would have been more useful for our application. However, we simply did not have the skillset to make an informed decision on the topic. A better approach would have been to invest more time in learning how to make a good choice at the beginning of the project. Ultimately, this approach would have saved time overall and could have resulted in a better final product. This learning piece is broadly applicable to future projects—a good approach is always to scope out exactly what you don't know before beginning any work.

Gamification

Similarly, we had challenges with abstraction of the complex process of protein folding into a game model and rushed forward with the first model that we all understood. Ultimately, this occurred because of the previously mentioned knowledge and communication gaps in the context of a short project timeline. However, a better approach would have been to do more experimentation with different game models at the beginning of the project. This would have allowed us to choose a model which was optimal in terms of technical challenge and user experience. Once again, understanding the scope of what we didn't know would have been useful here.

In general, our approach should have been to "front load" the project, rather than set ourselves up to have to do all of the important work at or beyond the halfway point. We may have been challenged by steep learning curves and large knowledge gaps, but we eventually proved to ourselves that these things are completely surmountable. Therefore, we could have gotten through them at the beginning of the project to be more successful at the end and to produce a better product.

Contribution Breakdown

Project contributions are as follows:

Christian

Christian was responsible for all high-level concepts, including gamification and concepts involving proteins. He worked closely with Ding to ensure that these concepts translated to the game properly and that the overall design/flow of the game made sense. Additionally, he created the presentations (including providing guidance on scripts) and wrote this report.

Ding

Ding was responsible for all 3D animation, designing the game UI, and implementing the scoring function and level selection screen. He built the game engine and created the algorithm which converts online protein information into blocks. Additionally, he provided feedback on presentation slides and scripts.

Zhong

Zhong's primary responsibility was to implement the property database and have it work with the game engine to map these properties to 3D. He assisted Ding with other coding pieces, including the game engine itself.

Future Work

Continued work on PeptiBlocks is contingent on the outcome of a conversation between group members that has not yet occurred. However, if work does continue there are several pieces which need to be improved and/or implemented in order for the game to be usefully playable.

Minor implementation pieces include real-time scoring, next block preview, and art design. These don't prevent the game from being playable, but they would make it much better in terms of user experience and accessibility of play. To this end, we would also need to include a built-in game tutorial. This is something that we could do relatively easily with the current state of the application.

Setting up a server for the app is the next *major* step for improving the functionality of PeptiBlocks. This server would allow us to store scores and user profiles and receive puzzle submissions from users. With a server we could implement a scoreboard for users to track progress and expand the number of levels we could offer. Most importantly, a server would allow us to store submissions and, further, to analyze those submissions in order to reconstruct final protein structures. Ultimately, this is the goal of PeptiBlocks so it is an essential step in app development.

If we were to effectively do these things then a long term view would also require us to validate the app by using a test set of proteins and determining if the game dynamics can predict their known structures. Following that, we would need to partner with a/some research group(s) to get protein puzzles for users then launch the app on the Play Store for users.

References

1. Berg JM, Tymoczko JL, Stryer L. Biochemistry. 5th edition. New York: W H Freeman; 2002. Chapter 3, Protein Structure and Function. Available from: http://www.ncbi.nlm.nih.gov/books/NBK21177/
2. Qing Li, Li Yi, Peter Marek, Brent L. Iverson (2013) "Commercial Uses of Proteases: Present and Future" *FEBS Letters* 587(8):1155-1163
3. Khatib, F. *et al* (2011) "Crystal structure of a monomeric retroviral protease solved by protein folding game players" *Nature Structural & Molecular Biology* **18**:1175–1177
4. Venayak N., Anesiadis N, Cluett WR, Mahadevan R (2015) "Engineering Metabolism Through Dynamic Control" Current Opinion in Biotechnology, **34**:142–152
5. Zhuang K, Yang L, Cluett WR, Mahadevan R (2013) "Dynamic strain scanning optimization: an efficient strain design strategy for balanced yield, titer, and productivity. DySScO strategy for strain design" *BMC Biotechnol 2013*, **13**:8.
6. Khatib, F. *et al* (2011) "Algorithm discovery by protein folding game players" *PNAS* **108**:47