DreamBook: Physial E-book

YOU-YI JAU, University of California, San Diego, USA
PANKHURI CHOUDHARY, University of California, San Diego, USA
GAURAV PARMAR, University of California, San Diego, USA
KEVIN BARRON, University of California, San Diego, USA
HARISH DUWADI, University of California, San Diego, USA

The paper presents a digital book with the feeling of physical book. While digital books like iPad brings convenience to reading, they fail to reproduce the feeling of reading on physical books. Our work brings the feeling of flipping pages for reading. The user flips the real paper to navigate through the book. At the same time, the content of books is stored digitally and projected to the paper. This leverages the benefits of digital books. Through testing and evaluation, this work claims good user interface for reading digital content. The final system brings possibility of a portable device with features from both physical and digital books (see figure 1).

Additional Key Words and Phrases: Human-computer Interaction (HCI)

ACM Reference Format:
You-Yi Jau, Pankhuri Choudhary, Gaurav Parmar, Kevin Barron, and Harish Duwadi. 2018. DreamBook: Physial E-book. 1, 1 (December 2018), 18 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

With the advent of immersive technology, devices are becoming smart ubiquitously. This seamless integration of connected devices has increased efficiency, while still retaining the comfort of maintaining old habits and lifestyle. In this paper, we discuss the disjoint experience of reading a physical book and an eBook. The latter has been around for many years, but it has been unable to obliviate physical books. Our aim is to develop a solution that would serve as a cross between the two and make the reading experience more ubiquitous.

To do so, we work on designing an interface that make reading an electronic book become more hands-on. By engaging in user centric prototyping, we gathered feedback on what would make the experience more coherent between the two popular ways of reading today. From the learnings gathered, we develop the first version of our product, DreamBook, that we believe would retain the favorable attributes from both of the interfaces.

In the remaining sections of this paper, we describe our motivation for aiming to improve the reading experience, the design process that led to this version of DreamBook and the features we implemented. We also discuss the features that we envision and plan on incorporating in the future, and other possible embodiments of this idea.

Authors' addresses: You-Yi Jau, University of California, San Diego, 9500 Gilman Dr, La Jolla, CA, 92093, USA, yjau@eng.ucsd.edu; Pankhuri Choudhary, University of California, San Diego, La Jolla, USA, pachoudh@eng.ucsd.edu; Gaurav Parmar, University of California, San Diego, La Jolla, USA, gparmar@ucsd.edu; Kevin Barron, University of California, San Diego, La Jolla, USA, k1barron@ucsd.edu; Harish Duwadi, University of California, San Diego, La Jolla, USA, hduwadi@ucsd.edu.

XXXX-XXXX/2018/12-ART $15.00
https://doi.org/0000001.0000001

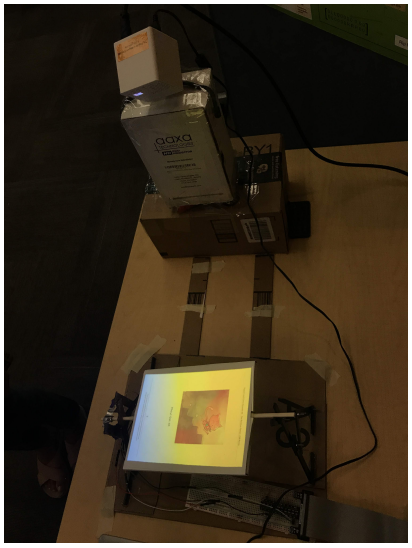

Fig. 1. Final design.

2 MOTIVATION AND BACKGROUND

Physical books have been the primary source of knowledge for millennium. Textbooks and other physical books, however, have many inconveniences associated with them. Such inconveniences include having to spend time in going to a store or library to pick up a book, having to pay for books that depreciate tremendously, or waiting multiple days for a book to be shipped.

With the introduction of e-Books, many of these inconveniences have been resolved. Now we can easily get access to the book of our choice in a matter of seconds and with the convenience of not having to leave our home. By choosing e-Books we not only save time, but also money; e-Books are comparatively cheaper than physical books. Not only do e-Books save money and time but they also have many convenient features such as the ability to bookmark, quickly search and find keywords and many other tools that add to the user's experience.

Although the benefits of e-Books are almost universally known, people are still using physical book instead of reaping the benefits provided by the e-Book. When asked for the reason of sticking with physical books the majority of readers respond that they like the feeling of touching a real book and flipping pages. This unique experience cannot be replicated by e-Books. Many readers also stated that they prefer annotating books as opposed to using the annotation feature provided by e-Books.

The idea of creating a book that has all the features of an e-Book with the feeling of a physical book is not a new idea. When researching this topic we find many research papers that discussed some variations of this idea. Flexpad, FoldMe, and PaperWindows are some of the variations of this idea. All of these in some way display a screen in a physical handheld paper or paper like object. Flexpad is an interactive system that uses a depth sensing camera and projector to display the screen onto an actual physical paper or a foam [3]. The depth sensing camera in FlexPad is used to detect the page fold and transform the display accordingly [3]. FoldMe is an interactive foldable display device, which has a screen that is foldable along predefined hinges [2]. The research paper for FoldMap discuss the different design patterns of their device that was based on the location of the hinges; one of the variation of the design is called Book device that mimics a book [2]. PaperWindows is a windowing environment that also uses a physical paper, but it simulates the physical paper as a digital paper

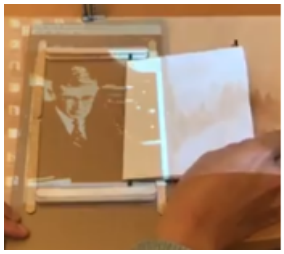

Fig. 2. Initial design.

Fig. 3. Version two of the one page design.

Fig. 4. One page design with three rows of magnets to improve page stability.

[1]. This prototype uses computer vision to sense the user's interaction on the real page with computer vision thereby making the actual page as an input device [1].

Although all of the above ideas provide users with access to books and are tangible devices, they do not solve the problem completely. FlexPad was intended for an applications like analyzing data and video slicing [3]. It doesn't have a feature that would help us detect the page turn to change the projected display. Also, since this device has fixed component it is not portable [3]. FoldMe is a portable device that has one design pattern that mimics an actual book, however the device doesn't have an actual page flip mechanism that can be used for page turning [2]. The user must use a touch screen to turn a page making it similar to other e-Book devices like kindle. PaperWindow uses an actual paper to augment the screen of a computer [1]. Users are able to read a book by touching and reading through an actual paper, however this device also doesn't have a page flip mechanism. This device projects a screen onto a paper, and page turning would also be a touch mechanism instead of an actual page flip.

All the devices and research prototypes in the market are able to solve only some of the problems that DreamBook addresses. DreamBook is a portable physical device that has the features that e-Books provide to users with the addition of an actual page flip mechanism to turn the pages of the book and provide the user with a similar experience to reading a physical book.

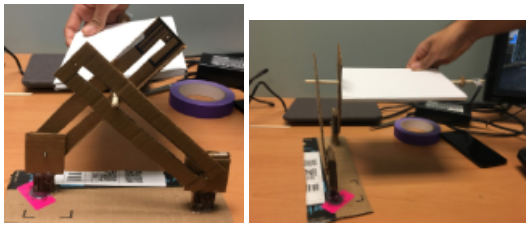

Fig. 5. Version three of one page design. The arms are no longer motorized and are moved manually by the user.

Fig. 6. Version four of one page design. The central axis of the page is fixed.

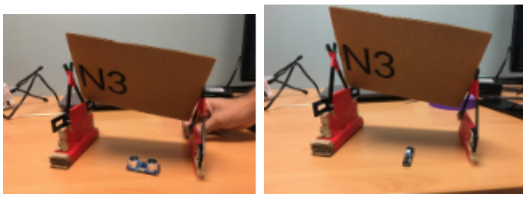

Fig. 7. One page design with two different methods of page detection. The left image uses a sonar sensor while the right image uses a light detector.

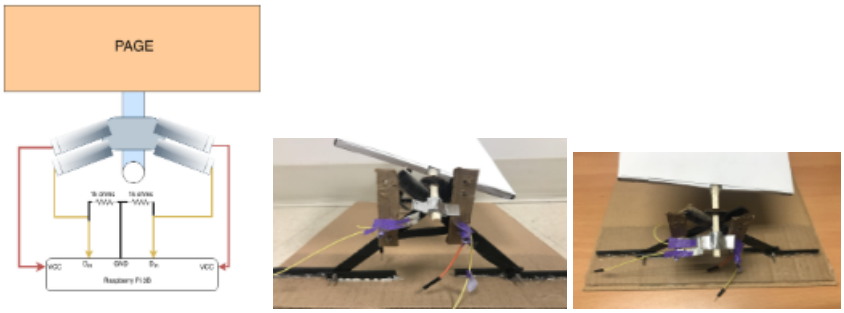

Fig. 8. Final implementation for page detection. This is a technique called "closing the circuit".

3 DESIGN

The design process for DreamBook was an ongoing effort throughout the duration of the quarter. The majority of the work was on the hardware end as the medium in which DreamBook was to be implemented in changed as well as how delivery would be accomplished in a given medium. In this section of the paper we will first discuss

Fig. 9. Final version of DreamBook.

the design of the hardware followed by software. These tasks were done in parallel but because for the most part there wasn't an immediate dependence between the two we have chosen to discuss them separately.

The first idea for implementing DreamBook was to use HoloLens to implement an AR application. The HoloLens would be used in conjunction with a blank "book" on which pages would be displayed. This would allow for portability and easy navigation as the surroundings could be used for book selection, going to bookmarks, taking notes etc. This idea, however, was quickly abandoned as the technology had limitations, mainly in the proximity and optimal distance for registering user actions which for our application was far too large. Because of the limitation we decided to explore other alternatives, mainly in using a small projector. We settled on using a Pico Projector to display pages on the book along with a Raspberry Pi for processing user input and instructing the projector what to display. The next step in the design process was creating the physical book on which projections would be made and how users would interact with the book.

Since we were going to be using a projector we had to create something that would act as the canvas for the projector. In this there were many questions but the biggest was how to capture the feel of a real book. Quickly we realized that the biggest differentiator between a real book and an E-Book was the ability to physically turn pages in a real book. The next realization was that the book that we created had to be able to "match" the number of pages in any given book so that our pages don't run out before the book ends. This of course isn't possible, so it meant we had to design something that in a sense mimicked having an infinite number of pages.

Our first design, as seen in figure 2, consisted of a single page on which the projector would display one page at a time. The page would be placed in between two rails that ran at the top and bottom. The rails were used in conjunction with four pegs in the corners of the page to guide turning of the page. As the user would turn the page the pegs would slide across the rails. This allowed for actually turning the pages but it was done in a manner that was not natural as the user had to essentially pull the page outwards in a horizontal manner before actually flipping. This initial sliding of the page made for an unnatural experience which led us to having to redesign the book in such a manner that page turning was done along some central axis. Around this time of testing we also got user feedback that it would be nice to have two pages rather than just one. This feedback led us to splitting design into two camps: one page design and two page design.

The idea with the two page design was that at any given moment two pages of the book are being displayed, just as in a regular book. This contrasts to our original design which only allowed for one page to be displayed at a time. Another important detail in the two page design was that the left and right page had to be independent of each other such that turning one doesn't cause the other page to turn, as this would just be a one page design with horizontal projection of two pages. Capturing such behavior turned out to be very difficult and in the end we could not deliver a prototype to give users to test. We did, however, come up with ideas that at a high level captured this behavior. The design consisted of having two intersecting pages and having the axis of rotation along the intersection of the two pages. This would partition the two pages into halves so the user essentially

has 4 pages. Two pages was the minimum required to implement the functionality we wanted but it posed the following problems. The first was keeping the pages stable. Having the sheets along a center axis would cause them to bend along this axis downwards. The only way of counteracting this was to put up "walls" to hold up the pages. This, however, would then create the problem of not being able to flip pages continuously as if there was a boundary holding up the page the next page wouldn't fit and get "reset" for future page turns. Having the pages "reset" was also a problem all together. To illustrate what is meant by reset imagine the original state of the four pages. There are two on each side. If a user were to turn the page forward, or from the right side to the left, the resulting configuration would be having three pages on the left and one on the right. "Resetting" refers to somehow balancing this out and making the bottom page on the left side move over to the right side. The best idea we could come up with was having each of the two main pages rigid such that turning the top portion would cause an equal turn in the opposite direction of the bottom portion and thus always maintaining proper page balance. This idea was never implemented, however, because there was no way to keep the pages stable without a wall but such a wall would inhibit full range of motion of page turns.

Because the two page design was not feasible we had to settle on pursuing the one page design. The one page design was iterated upon multiple times but the second version, after the initial idea of top and bottom rails, was designed in parallel of the two page design.

Version two of the one page design consisted of having a central vertical axis within the page and this entire axis would be placed on perpendicular arms of a servo motor, as seen in figure 3. The central axis would solve the problem posed by the first version of the design of unnatural page turning. The servo motors were incorporated because we wanted the page to be as low as possible to the surface of the base but in order to turn the page properly space had to be created between the base and the page. Thus when the user would start to turn the page the servo motors would begin raising the page and after a turn was completed they would return to their original state. This design also posed some problems. The first was that the servo arms prohibited fixing the central axis to any point of reference so the page could be removed at any time. The second problem was that timing motor speed was very difficult to calibrate and if the servos rose too quickly the page would fall out and similarly on the descent the page would move around and lose central alignment. We tried tackling the second problem first as the solution seemed easier. The next iteration of this version of the design would include magnets, as seen in figure 4 in the page and an electromagnet in the base. The electromagnet would be in between two strips of magnets and the repulsive forces would ideally cause the page to resist losing central alignment. Then after each page turn the electromagnet would change polarity to continue providing a repulsive force. Although in principle this would solve the problem it was never fully implemented as such a design would be of little value if the first problem was not solved. Keeping the page from falling out was not possible unless the central axis was secured to something, but this would prevent the servo motors arms to work properly. This obstacle would lead us to a third version of the one page design.

Since the biggest problem with using servos was getting the timing correct on lifting and dropping the page our third design tried using the same concept of a lifting and dropping mechanism but without being motorized. This can be seen in figure 5 where the main idea is that the arms simply guide the user in the motion of lifting and turning the page as the rails in the first design did but with the advantage that the page would not be removed. The motion of turning pages was fluid in this design but the problem was that in its resting state the page could move side to side which made for an unnatural user experience as pages in books don't move.

Our fourth design was much more representative of the final version. We didn't want to compromise on the central axis of the page being the pivot point as removing this would deviate from the feel of a turning a page in a book. Instead we decided to fix the position of the axis, as depicted in figure 6, and permanently maintain it elevated such that there is clearance for the page to turn without the need of servos. The only difference between this design and the final was that in our final design we opted towards having the page angled similarly to how a podium would be angled for easier reading. In this design the page would be kept parallel to the base.

With the orientation and turning mechanism of the page established the last design choice to be made was on how page detection would be captured. Addressing this was a process that started as early as our first iteration of the one page design but we have refrained from discussing it until now as sensor implementation for the most part would have been the same regardless of which design was chosen. More specifically, the choice of which sensor or technique to use was universal enough to be applied to any of our designs, even the two page design. Throughout the design process there were three techniques for page detection that were explored: use of sonar sensors, use of light detectors and use of aluminum in a process we will refer to as "completing the circuit". The last approach was the one that made it into the final design but we will first discuss the first two approaches and their eventual shortcomings.

As mentioned the first approach was to use sonar sensors. In this approach the idea was that sonar sensors would be placed on the base, as seen in figure 7, under the page to measure the distance and change in difference to determine if the page had been turned. Initially two sensors were going to be used, on both sides of the page. In this implementation the distance between each sensor and its respective half of the page was measured and the difference was computed to detect if a page had been turned forward or backward. This approach in practice, however, did not work because of a limitation in the capabilities of sonar sensors. As sound waves are conical in nature sound emitted from the sensors would interfere with the readings of the other sensor which created "noise" in the data our algorithms would work with. The "noise" was further intensified when the page was mid turn as sound would be emitted from one sensor and bounce of the page directly into the other sensor. We then tried using one sonar sensor but again when the page was turned the distance returned by the sensor was not the distance between base and page but base-to-page-to-wall and back. The only way to counteract this would be to build boundaries around the page but we found this solution to compromise the experience of turning pages and creating an algorithm that accurately determined a page turn was quite cumbersome.

The next idea was to use light detectors. In this approach a source would emit light to the light detector and placed in such a way that when a page turned the page would block the source to indicate that the page had been turned. This approach was problematic since when indoors there was bound to be light pollution which would interfere with proper sensor readings. We then tried using the light detector in a similar position as the sonar sensors, directly under the page, but again detecting any change in lighting was difficult to light pollution. The second attempt of using light detection is seen in figure 7. In this approach our algorithms could not properly detect page turns and thus we had to abandon this idea as well.

As traditional sensors continued to be problematic we decided to pursue a simpler method for page detection. We came up with a method that we call "completing the circuit". The idea is that in detecting page turns we only need a signal that keeps track of turning pages forwards and another backwards. The technique of completing the circuit can be viewed as page turns pressing a button, which completes a circuit and allows for a signal to propagate from one place to another. The design consists of placing aluminum strips orthogonally to the central axis on which the page turns such that these strips come in to contact with two other strips that are stationary. When the three strips come in to contact they act in a similar way as a button would when pressed and complete the circuit. For further understanding refer to figure 8. This contact allows for a signal to be sent alerting the system that a page has been turned, either forwards or backwards.

As a conclusion to the design of the hardware in DreamBook we will quickly recap all the features in the final design. The system processing the signals and running our code is a Raspberry Pi 3 and the projector being used is a pico projector. As our current iteration is a proof of concept the material being used for the page itself is cardboard covered with white construction paper. A chop stick was placed vertically through the page to act as the axis on which the page turns. The page and chopstick rest on an acrylic support on both sides that suspends the page above the base to allow for clearance when turning pages. Strips of aluminum foil are placed on the chopstick as well as attached to the acrylic to implement the signal detection of turning pages. This final design is seen in figure 9.

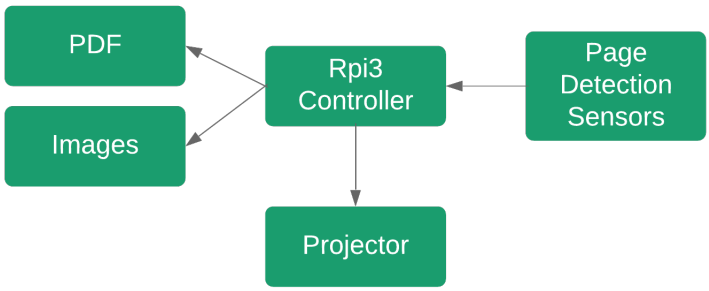

Fig. 10. System overview.

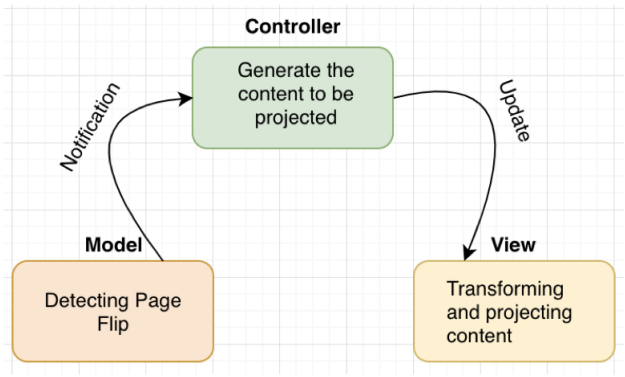

Fig. 11. MVC design pattern.

Of course DreamBook is a hardware and software implementation so we will now touch on the software design. Similarly to the sensors our algorithms were for the most part independent of hardware design and was done in parallel of hardware development. The software architecture was broken into three pieces based on functionality. These functionalities were applying a transform to the projection such that it would fit on the physical page, changing the page being projected based on user input and of course the signal processing for page turns. Implementation details will be further expanded on in the next section of the paper but at a high level all three parts of the software were implemented in Python and using the libraries numpy, scipy, openCV and pdf2images.

4 SYSTEM DEVELOPMENT

In this section, we describe the architecture, the technology and features in our system. The high-level architecture is described in section 4.1, and the technology is discussed in hardware and software part in section 4.2. In the software part, python packages are introduced and described. In section 4.3, modularization is discussed. Also, the calibration module and page flipping detection module are described in detail.

4.1 Architecture

We use Raspberry Pi 3 Model B+ (Rpi3) as the controller to our system. Rpi3 controls the pdf files and images, accept the signals from page detection module, and displays the content through the projector. The architecture is shown in figure 10.

4.2 Technology Used

4.2.1 Hardware. The hardware components used in our system are the Raspberry Pi 3 Model B+ and the AAXA pico LED projector.

Raspberry Pi 3 Model B+ is chosen as the primary platform for doing the computation because it is readily available and has a large online developer community. The platform also supports running a complete graphical based linux operating system and has multiple cores to enable different modules of our system run in parallel.

AAXA Pico LED projector is chosen primarily because of its small form factor. Traditional projectors are large and would make the system bulky. The chosen projector has a portable cuboidal design, weighs 6 ounces, and has an inbuilt battery that lasts approximately 2 hours. These features make the Pico projector an ideal choice for DreamBook.

4.2.2 Software. Python 3 (3.5) is used to implement the functions. The following packages are utilized for development.

numpy is utilized mainly to create arrays that can communicate with opencv-python.

pdf2image is utilized to read in a pdf file, and convert to a list of PIL object. This package depends on a low-level package "poppler" for displaying pdf file in GNU/Linux.

opencv-python (cv2) is utilized to display the images. In the beginning, cv2 displays a black background. After the pdf file is imported with pdf2image. The list of pages are transformed into cv2 images. The cv2 images go through transformation matrix. The transformation matrix is of size 3x3. It is computed in calibration module. The detail of computing calibration is described in section 4.3.2.

scipy is utilized to acquire the nearest 2D position in the calibration process, using KDTree.

4.3 Features

The software is developed regarding to The Model View Controller (MVC) design pattern. The modularization is discussed in section 4.3.1. 2 modules, calibration and page flipping detection, are discussed in section 4.3.2 and 4.3.3. .

4.3.1 Modularization. The followings describe each module in MVC design pattern. See Figure 11

Model Our model contains the user interaction part: page detection and keyboard input. Page detection receives input signals from GPIO. After denoising and processing, the notification of page turning is sent to the controller. Keyboard input is realized by imshow in cv2. We use keys to select pdf file and jump out the calibration process.

Controller The Controller contains the main part of the system, including storing the content, receiving notification from the Model, and sending signals to the View for display. To store the content, we read in a pdf file through submodule "pdf2img" and convert to list of cv2 images. To calibrate the scene, we get the position of page and compute the transformation matrix. To display the content, we transform the images with the calibration matrix. To receive the signals from the Model, multi-threading is used. There is one thread running in the background to detect the signal from the user. When the action of page flipping is completed, action is stored in a global array. There is one main while loop running for display. The main while loop checks if any action is stored in the array. If there is, action, either flipping forward or flipping

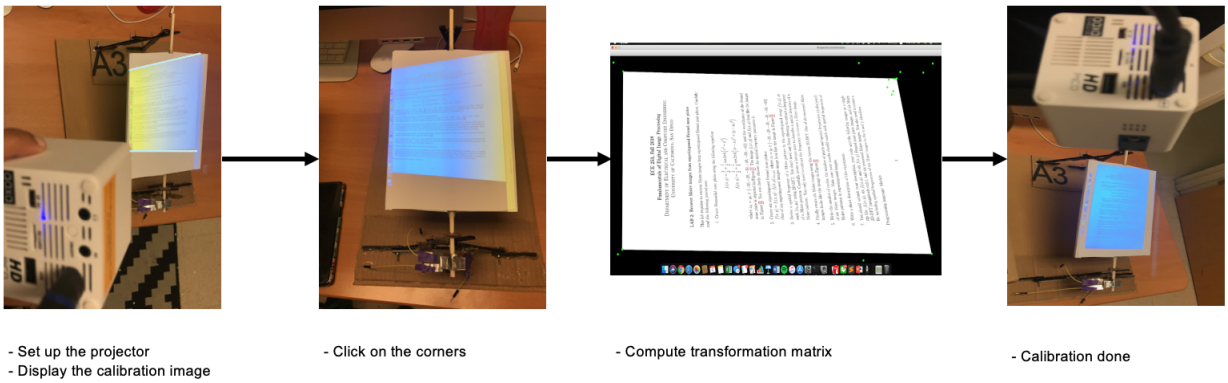

Fig. 12. Calibration flow.

backward, is achieved by changing image for display. If the page number is out of range or no action stored, the Controller sends the same image for display.

View The View take in charge of showing images to the user. In our system, this part is realized by cv2 using imshow.

4.3.2 Calibration. We use calibration submodule to fit the image onto the targeted plane. In the beginning, the display plane is in sheared shape. The reason is that the projector is not perpendicular to the projecting surface. After calibration, the image can fit on the page. The following steps describe how to calibrate the projector and generate transformation matrix. (See figure 12)

Set up projector We set up the projector on the left side of the book base in 30cm, at the height of 50cm. The distance ensures that the page is fully covered by the projection, and brings up good resolution.

Display the calibration image We read in the selected pdf file and process using 'pdf2img' submodule. Then, the first image is displayed for calibration.

Click on the corners When the calibration image is displayed, we can see the corners of the image. The goal is to fit the 4 corners to the right position. We use the mouse to click when mouse is at the desired position.

Compute transformation matrix The nearest corner is chosen and the calibrated position is updated. The corner is represented by vector (x,y,1). Using the 4 corner-4 corner correspondences, we can compute 3x3 transformation matrix. The nearest corner is generated by function in scipy. The transformation matrix is computed using cv2.

4.3.3 Page flipping detection. We use hand-crafted switches to detect the motion of the page.

Design of the switch We put one switch on each side, so that we can tell flipping forward from backward (figure 13). A switch is made of 2 pieces of aluminum foil. The 2 pieces are detached with each other, so it is open loop. In open loop, the GPIO has voltage = 0. When the wing of page stick touches the switch, it forms a closed loop. This makes the GPIO has voltage = Vcc (see figure 14). The GPIO voltage from both side are read by Rpi in 100Hz.

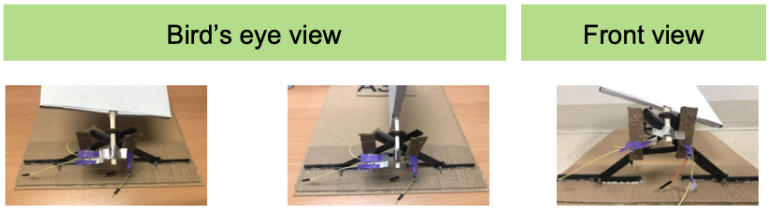

Fig. 13. Page flip detection.

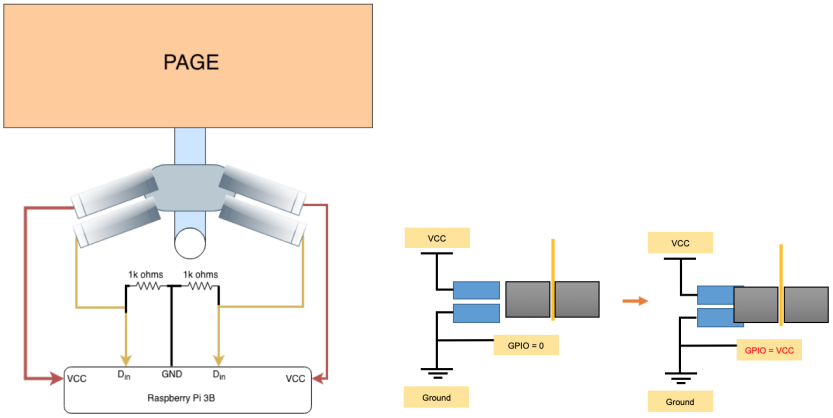

Fig. 14. Page flip circuit design.

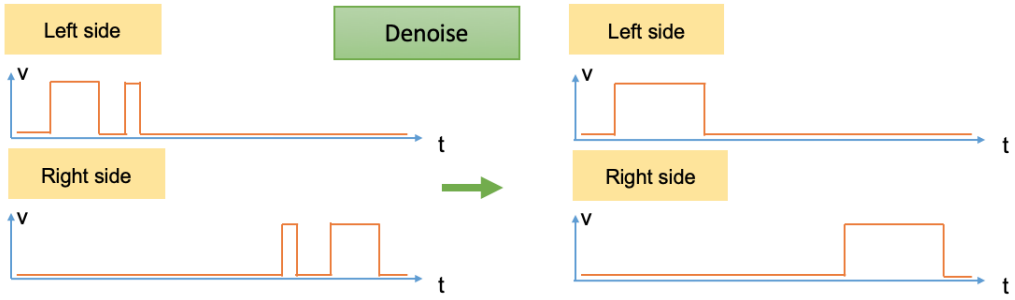

Fig. 15. Flipping signals denoising.

Denoising The signal read by Rpi contains noise, including jitter between 0 and Vcc. We denoise the signal by smoothing (see figure 15). After that, 2 streams of signals (right and left switch) are processed to decide which action to make (see figure 16).

Fig. 16. Flipping action decision.

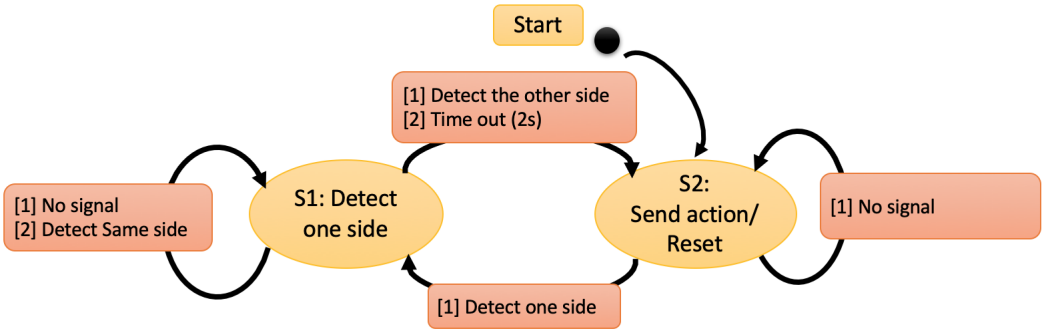

Fig. 17. Flipping finite state machine (FSM).

Action decided Action decision is realized by finite-state-machine (FSM) (see figure 17). In the beginning, it is in state 2(S2). When one switch is touched (either left or right), it enters state 1(S1) to wait for signal from the other side. At this moment, if the same side of signal is detected, it remains in state one. If the other side is detected, it enters S2 and decide action. If it is in the order of left-right, action is flipping forward, and vice versa.

5 TESTING AND EVALUATION

Testing DreamBook centered around iterative usability testing of the hardware and in house testing of the software throughout the duration of development. Ideally we wanted to test software and hardware together in user testing but because of a dependency on hardware the software could not be properly user tested until the hardware and physical design was complete, and thus a compromise had to be made.

Testing of the hardware and design was done throughout the entirety of the quarter, from ideation through implementation. Our first step was creating ideas for implementation and putting them up to the test by asking potential users how they felt about possible use cases or implementations. The people we asked were of course ourselves first but then we also asked a few friends and our TAs. Based on the feedback they gave us we decided to implement the design ideas, mainly the preliminary one page design as discussed in the design section. Once this was implemented we tested in house to make sure the design functioned as desired and to practice the roles in a WOz prototype.

Our first documented user testing session was in week 5 when we tested our preliminary one page design with two users: Janet, one of our TAs, and Ted, one of our friends who had experience with HCI. We implemented a WOz prototype that allowed the user to use the physical design to turn pages but the projected pages would

be changed by one of our members. The feedback we got from these two users was the flipping forward was intuitive but flipping backwards was slightly more difficult. Another suggestion was that it would feel more like a real book if two pages were displayed at a time. The feedback did not indicate that the method of displaying pages via projection was an issue.

The first set of user testing was concluded with iterating off of the feedback to create a redesigned one page design and a new two page design. We had a second opportunity of user testing and feedback after a few sessions of ideation in lecture in a following week. Here we did a presentation of our design ideas to get feedback on which implementation people preferred. The feedback didn't sway one way or the other so we proceeded with trying to implement the ideas as a physical demonstration might help users' decision making.

Unfortunately because of design constraints and timing we were never able to fully implement the two page design so it never made it to another round of in house or user testing. The new one page design that worked around a pivot rather than a set of rails was completed. After testing in house we presented the design to one of the other teams in the class during a discussion section. This user testing focused solely on the interaction with the device so no WOz prototype interface was created to display projections. This testing was more successful as there were no issues in turning pages forwards or backwards and every action was reported to be intuitive. We did receive minor feedback however, like placing the page at an angle to make reading easier from a seated position. After this session of user testing we had completed our cycle of usability testing and had come up with a final design for the hardware.

As mentioned before all software testing was done in house by members of the team. During development all software testing was done as unit tests for individual components of the system. The three main components of the system that required testing was page turn detection, changing pictures being displayed by the projector and applying a projection transform such that pages would fit on the physical page. Page turn detection required various tests for the various sensor we had experimented with. The common theme was that in testing sensors we would print output to the terminal when an action was taken by the system when a signal was detected by a sensor. This approach was used for sonar sensors, light detectors and ultimately in circuit completion detection. The test was passed when various page turns in either direction printed the correct action without interference from noise collected form sensors. To test that projected pages turned forward or backward correctly we hard coded a signal and the output to this signal was observed. The test passed when the projection sequence of page turns mirrored the expected output. Testing of projection transforms was done by projecting an ordinary image on a wall and adjusting the image to transform such that the projection fit within a designated and marked portion of the wall. Once the image correctly transformed into this area the implementation was considered to having passed the test. Once unit testing was completed we merged the implementations into one project. Testing this complete implementation was done through testing the entire system with the hardware. Each of our members had the opportunity to use the device and flip through pages to ensure that projections were shown correctly and page turns resulted in correct updates of projections.

Based on our internal testing our system works as expected and as we had hoped it would at the beginning of the development cycle. There were some features that we did not get to implement because of time constraints but of the core features we did implement they all worked as expected. Users are able to successfully power on the device to begin reading a book and through flipping pages the user is able to navigate across pages of the book.

Our original idea for DreamBook would be that it would be a hybrid between physical books and E-Books to bring the benefits of these two ways of reading. Our design captures all of the benefits that an E-Book would provide with the only current limitation is the lack of an abundant library of books. This restriction however was apparent from the onset as we do not have the licenses to host all the books and instead we make use of pdfs provided to us in the course readings of this class. The idea, however, of capturing the benefit of an E-Book in our opinion has been fulfilled. Capturing all the benefits of a physical book is still something that needs to be

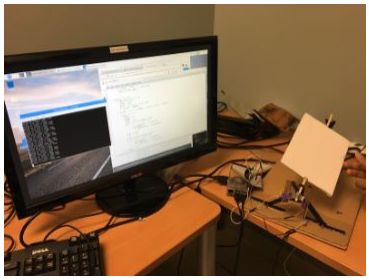

Fig. 18. Testing sensor reading by printing to terminal.

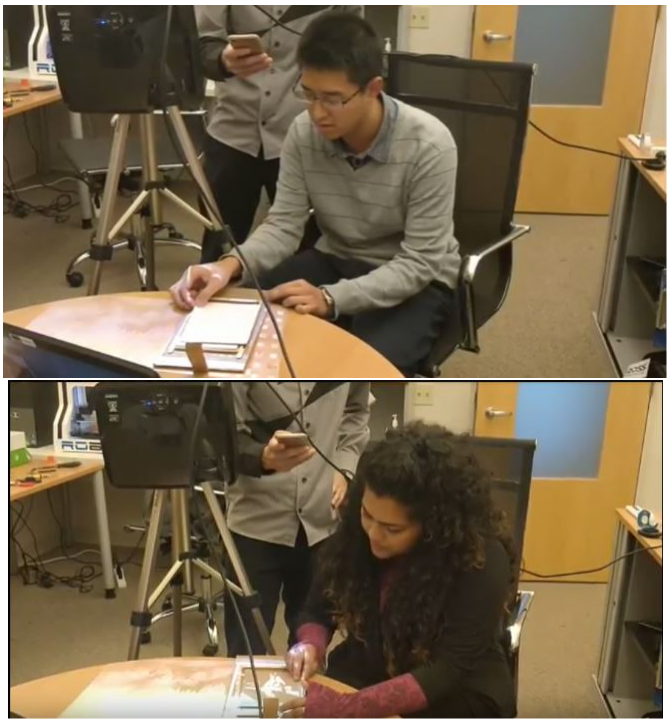

Fig. 19. User usability testing with WOz prototype.

worked on. Although we enjoy reading on our device we acknowledge that in its current form there are elements that are missing. Page turning and navigation are the main features we wanted to capture and by this metric we were successful but there are other aspects like portability and note taking or annotating that we have not been able to implement yet.

6 COLLABORATION

The team consisted of five members. Pankhuri and Youyi were graduate students, pursuing an MS in Computer Science and Electrical Engineering respectively. The three undergraduate students were Gaurav, Harish, and

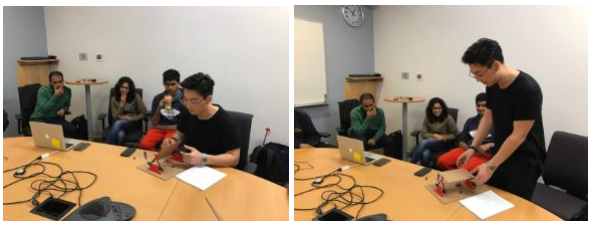

Fig. 20. User usability testing with low fidelity model.

Kevin; each of which were pursuing a BS in Computer Science, Computer Engineering and Computer Engineering respectively. From the start of the project our approach to all of our tasks was to work together in subgroups and only when there was left over work to do it individually. It is also worth mentioning that although we did divide into subgroups all members would be present in the same space for immediate feedback from the other group when needed.

The approach of working in two subgroups originated when ideating for what our project would be. After a brainstorming session we created a short list of possible ideas but then narrowed them down to two ideas: Dreambook and an AR application that would teach users how to play the piano. Since we had two ideas we found it natural to divide into two subgroups to further research and develop both ideas. Specifically, Pankhuri, Youyi and Harish developed the idea that would become Dreambook while Gaurav and Kevin developed the AR piano teacher idea. After each subgroup had a weekend to develop the ideas we regrouped and presented them to each other. This was followed with some debate before coming to a consensus to implement Dreambook. The process was so smooth that we decided to stick to it for the rest of the quarter.

In the design phase of the project Pankhuri, Youyi and Harish explored a two page implementation while Gaurav and Kevin worked on a one page implementation. Harish and Youyi came up with various designs for the two page version while in parallel Pankhuri would work on creating physical models of their designs. For the one page version Gaurav and Kevin took turns coming up with ideas and implementing each others suggestions. This approach mirrored what pair programming is in software development, allowing for both individuals to participate in both roles. At the end of the design phase both subgroups presented their ideas and as a team we came to the consensus of proceeding with the one page version. Finally, for the parts of the project that required use of PrototipAR Gaurav was in charge of being the "user" while Kevin would create the drawings and act as the "computer" in WOz sessions.

In the development process we again divided into two general subgroups, software and hardware, but tasks were slightly more individualized. Pankhuri was assigned the task of implementing the functionality of being able to change pages of a pdf given an input from the hardware. This task also included researching various methods for accomplishing the task, like working directly with a pdf or first converting it to an image and then moving from picture to picture. Pankhuri was also responsible for the associated unit testing. Youyi was assigned the task of developing code for being able to perform transforms of images to fit onto the physical page when projected. This task included research for libraries for camera calibration and projection mapping which would then be implemented using Python. The implementation also included learning how to use opencv for projection calibration of images. Youyi would later go onto merging this portion of the code with what Pankhuri wrote for changing pages in a pdf. The last task on the software end was done by Harish and included implementation of code for signal processing from the sensors we used and generating the appropriate output based on the signals which would act as input to the algorithm implemented by Pankhuri. This included implementing two versions which used light detectors and sonar sensors respectively. As discussed in the design section of the paper these methods of page turn detection were not used in the final version but the work was crucial in the

design and iteration process of our project. Hardware was handled by Gaurav and Kevin. Both Gaurav and Kevin worked through various iterations of the one page design and how sensors would be integrated. Gaurav was responsible for integration of light detectors and implemented the final version of hardware page detection using the aluminum strips as buttons. Gaurav was also in charge of laser cutting acrylic and writing the code that was used while the design included the use of servo motors. Kevin was responsible for putting together the various designs. This included creating the supports for the page, the page itself, and assembling or attaching the sensors used in various versions of the design. Kevin also assisted in debugging the problems with use of sonar sensors. Integration of the hardware and software was done by all members of the team, as well as in house testing.

The final portion of the project consisted of making the poster and writing this paper. For the poster Harish did the motivation and background, Gaurav did the design section, Youyi did the system development, Kevin did the testing, evaluation and conclusion section and finally Pankhuri did the future work and references portion. Additionally, we all contributed into editing each others sections until we came to a final version of the poster. The paper was written in individual pieces before being revised and transferred into Overleaf. The intro and conclusion was written by Pankhuri, the motivation and background was written by Harish, the design, testing and collaboration sections were written by Kevin and the system development was split into hardware and software between Youyi and Gaurav. The transfering to Overleaf was done by the whole team but mainly led by Youyi and Gaurav as they had the most experience with Overleaf and LaTeX.

Throughout the development of this project we made use of three tools to help with overall collaboration. To begin with we used Facebook Messenger as our main means of communication. Messenger was chosen as everyone on our team already used it regularly and there was no learning curve. To keep everyone on track and accountable for their work we made use of Github's project feature as well as the Google doc for weekly updates provided to us by the course staff. Github was used in place of Trello to keep track of tasks to be completed, tasks in progress and completed tasks. The weekly updates document on the other hand was used to quickly gauge what each member was working actively working on and provided additional details such as potential roadblocks that Github alone could not capture. The weekly updates also provided us with the ability to see how our timeline and deadlines would change as we would update the milestones and deliverables on a weekly basis for all remaining weeks in the quarter. For file storage we used Github for our code and our team Google Drive for any other files.

There were of course tools that we did not use and some that we used temporarily but eventually discontinued. The three common tools that we did not use were Slack, Trello, and Gantt charts. We did not use Slack as we already had Messenger for communication and our Google Drive for file sharing. Additionally Slack did not have the functionality of video calling which we made use of whenever someone could not physically attend a group meeting. We decided not to use Trello as Github already provided us with a card system for viewing current and completed tasks. Finally Gantt charts were not used as our weekly reports included milestones and deliverables that would be updated every week as the quarter progressed. Gantt charts of course would have been easier to visualize but we didn't believe this benefit outweighed the cost of having to use an additional separate tool. This coupled with the fact that we stayed in contact frequently enough to understand what work was left and what pace should be maintained is what led us to the decision of not using Gantt charts. There was one additional tool that we used in the first half of the quarter but later replaced and this was Dropbox. Initially Dropbox was used for file storage and also in place of a website as we had all of our administrative information in one main document. This was dropped because the majority of our team did not enjoy using Dropbox as editing files was not as simple as in Google Drive. We found Drive to be a better alternative and once we setup our Github repository there was no need for Dropbox as our repo had a section to include all relevant administrative information.

Another key feature of our collaboration was our meetings. For the first half of the quarter we decided to have regular weekly meetings on Thursday evenings for two hours but with the agreement to stay later if it meant

finishing up the tasks for the week. If there was work left over, which was usually the case, we would meet up on Sundays to finish. In the second half of the quarter we doubled the meeting times to Tuesdays and Thursdays, not including the time allotted to us during lectures and discussion. In addition, subgroups would meet throughout the week to continue working on their respective tasks. Initially meetings were held either at PC or the VR lab but once we were permitted access to CSE 3205 our meetings were always there.

In truth we did not have any issues or problems internally. The closest thing we had to an issue was when not everyone was able to meet up for a meeting. This was dealt with through two principles that we agreed upon as team. The first was that the person who missed a meeting was expected to video call either during or after a meeting for input on any decisions to be made and also to be caught up with what they missed. Additionally if someone was to miss a meeting and had work expected of them they were required to have it completed prior to the start of the next meeting. We do not attribute our seamless cooperation to luck, we took precautionary measures from the beginning. In our initial meeting we went over the things that we could and could not tolerate from team members. From the onset we established that missing meetings without adequate prior notice, being late to meetings and failure to finish tasks that were promised was not acceptable. We also realized the importance of having a good team dynamic so we made the effort to actually become friends and get to know each other outside of just our work. Weekly meetings were always started with "warm up questions", open ended questions presented by a different member each meeting. Examples are "where would you like to travel" and "what is your favorite/least favorite season?". These helped us get to know each other and also start the work day with some fun. Additionally for those of us that lived off campus we would commute together after meetings, which allowed for having conversations unrelated to school or work. The way we approached building our team and the selection of tools we used led to positive overall collaboration.

7 CONCLUSION AND FUTURE WORK

In this paper, we explored options to combine the benefits of reading a physical and an electronic book. We went through the various iterations of our prototype designs and described how the product evolved, outlining the challenges we faced and the resolutions made. We presented our version of DreamBook that aims to make the reading experience more ubiquitous, and also real user response for it.

We were limited in the features that we could build in the given time frame, for the purpose of this class. In the future, we want to integrate more useful features. DreamBook should have the ability to save bookmarks and open a book at the saved bookmark, in a ubiquitous way. We also want to have the functionality to highlight passages, make notes and save them for future reference. We want to make the design more compact and mobile. In the future, we could potentially use AR to project the book once AR gadgets become more usable at closes distances. Having an LCD display panel for the page would also be an interesting way to make today's kindle feel more like a real book.

Given the response of the users for our first version of DreamBook, we feel that once functionality is improved and design is made leaner and more stable, DreamBook should be able to capture the essence of a physical book while integrating the benefits of reading eBooks.

ACKNOWLEDGMENTS

We would like to acknowledge Janet Johnson and Danilo Gasques for their help throughout the quarter in brainstorming ideas and feedback on the prototypes. We would also like to thank our instructor Weibel Nadir for the class lectures and helping with the design process. We also acknowledge the other students in the CSE 118/218 classes for feedback provided during the discussion sessions.

REFERENCES

[1] Mark Altosaar Nikolaus Troje David Holman, Roel Vertegaal and Derek Johns. 2005. Paper windows: interaction techniques for digital paper. (2005), 591–599. http://delivery.acm.org/10.1145/1060000/1055054/p591-holman.pdf?ip=137.110.58.83&id=1055054&acc=ACTIVE%20SERVICE&key=CA367851C7E3CE77%2E30F2CC1B2E6C2B37%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1544763770_d1bb7ea3305561dc541458b65bc65cd1

[2] Wolfgang Kleine Mohammadreza Khalilbeigi, Roman Lissermann and JÃijrgen Steimle. 2012. FoldMe: interacting with double-sided foldable displays. (2012), 33–40. http://delivery.acm.org/10.1145/2150000/2148142/p33-kahlilbeigi.pdf?ip=137.110.58.83&id=2148142&acc=ACTIVE%20SERVICE&key=CA367851C7E3CE77%2E30F2CC1B2E6C2B37%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1544763744_c06dcae867cc31399adc148deab71056

[3] Jordt Andreas Steimle JÃijrgen and Pattie Maes. 2013. Flexpad: Highly Flexible Bending Interactions for Projected Handheld Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York USA, 237–246. http://delivery.acm.org/10.1145/2480000/2470688/p237-steimle.pdf?ip=137.110.58.83&id=2470688&acc=ACTIVE%20SERVICE&key=CA367851C7E3CE77%2E30F2CC1B2E6C2B37%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1544761063_3c8e5a5647ccfbb521fb5f19a041fadb

Received December 2018