김유준 | 20200130
kyujun02@kaist.ac.kr
이상헌 | 20200459
buaaaaang@kaist.ac.kr

CaperDoc: New Learning Platform for PDF Based Studies

1. 아이디어 제시 배경

기술의 발달로 아이패드 등 펜을 기반으로 한 여러 pdf 뷰어와 필기 어플리케이션들을 이용해 공부를 하는 학생들이 많아지고 있다. 우리는 수학 공부를 하면서 한 정리를 증명하기 위해 다른 여러 정의와 정리들을 참고하여야 했던 경험이 있다. 종이책과 달리 pdf 뷰어에서 많은 페이지를 한번에 이동하기 위해서는 수많은 스크롤을 하거나 페이지 번호를 기억했다가 입력해야 한다. 설령 북마크 기능이 있다고 하더라도 이름을 제대로 지정해 주지 않으면 역시나 나중에 원하는 위치로 찾아가기가 힘들다. 기술이 발전으로 우리는 더이상 두꺼운 전공 서적을 가지고 다니지 않아도 되지만, 여전히 위와 같은 문제점들이 생활속에 남아 있다.

우리는 "쉽게 공부할 수 있는 학습 기반을 만들자"라는 모토로 프로젝트를 진행했다. 다시 수학을 공부하는 상황으로 돌아가보자. 대개의 경우 교재에는 참고된 부분이 theorem N 과 같이 기호로 나타나 있다. 하지만 이 기호만을 보고 실질적인 내용을 기억하는 것이 힘들다. 특히 어느 정리가 몇번째 페이지에 나와있는지 조차 나와있지 않아 예전 정리들을 찾기 위해서 페이지 사이를 오가면 시간과 집중력을 낭비하게 된다.

이런 상황은 수학 뿐만 아니라 다른 과학 분야나 법률을 공부할 때도 비슷하게 일어난다. 한 현상을 설명하기 위해서 이전에 나왔던 수많은 현상들이 언급되고, 한 문장의 논리를 펼치기 위해서 이전에 나왔던 수많은 명제들이 다시 언급된다. 하지만 서적의 물리적 한계를 고려하였을 때 모든 디테일을 전부 다시 언급하는 것은 불가능하다. 그렇기 때문에 정리나 표 등에 번호를 붙여두고 그 번호를 참조하라는 식의 간접적인 언급만이 기재된다.

우리는 이에 대한 해결 방안으로, 정리들 간의 링크를 잘 만들어서 페이지 이동이 편리해 지도록 프로그램 개발에 착수했다. 우리는 사용자들이 이 프로그램을 통해 중요한 개념들에 대해

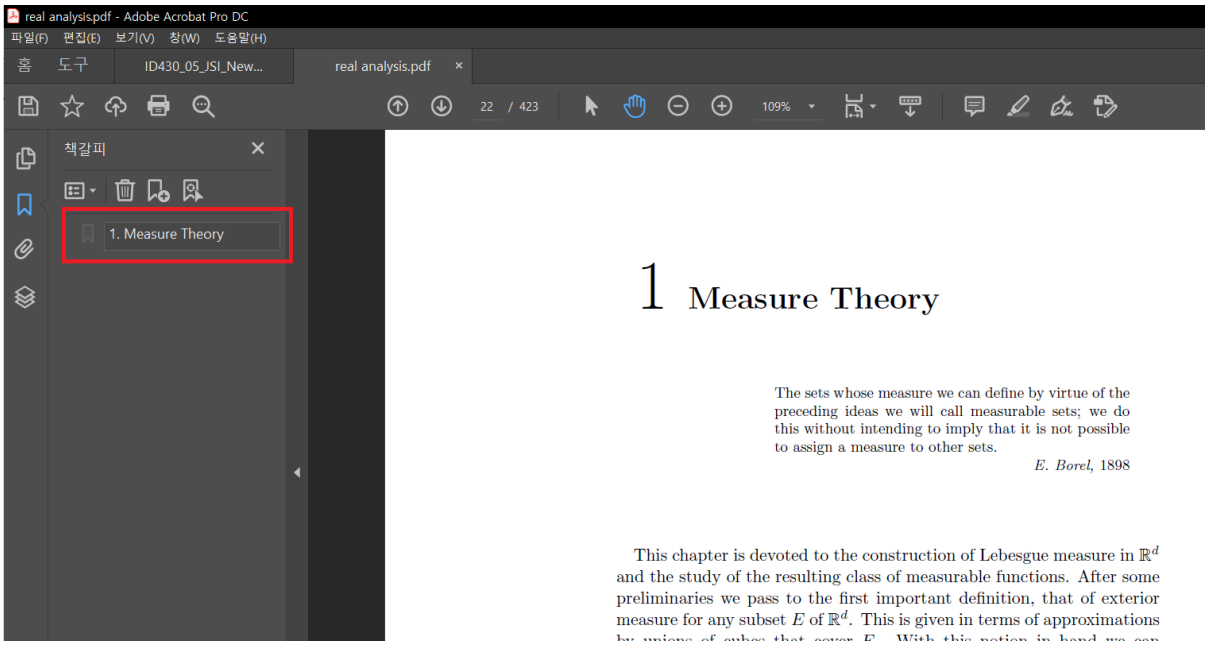
(1)Hierarchy 를 저장해두고 (2)이 Hierarchy 들 사이의 링크들을 저장하고 발전시켜 나가서 최종적으로 (3)학습 중인 PDF 에 있는 여러 개념들 사이의 상관관계를 보다 쉽게 이해하고 참조할 수 있기를 기대한다.

2. 관련 연구

우리가 구현하고자 하는 기능은 기존 프로그램의 책갈피 기능과 가장 유사하다고 생각했다. 때문에 현장에서 사용되고 있는 PDF 뷰어 및 필기 앱의 책갈피 기능을 살펴보았다.

(1) PDF Viewer: Adobe Acrobat Pro DC

Adobe Acrobat Pro DC(Adobe)는 가장 많이 사용되는 PDF 뷰어 중 하나이다. Adobe 에서 책갈피를 생성하기 위해서는 특정 페이지로 이동하고, 북마크를 추가하는 버튼을 눌러야 하며, 이때 사용자가 직접 북마크의 이름을 지정해 주어야 한다.



우리는 기존의 이 프로그램에서 두가지 문제점을 발견했다. 첫번째로 Adobe 에서 책갈피를 설정할 때 이름을 지정해주는 작업이 번거롭기 때문에 기본완성 되는 “제목없음”이라는 이름의 북마크가 만들어질 가능성이 높다. 하지만 이런 북마크는 나중에 어떤 정보를 담고있는지 알기가 어려워 북마크로써의 기능을 하기 어렵다. 두번째로 이 프로그램은 페이지 자체를 북마크 해주기 때문에 페이지에 어느 부분이 중요해서 북마크를 했는지 알수가 없다. 만약 사용자가 북마크를 통해 이 페이지로 넘어왔다고 하더라도 원하는 개념을 찾는데 또다른 시간을 소비해야 한다.

Caper Doc 은 이 두가지 문제를 독자적인 방법으로 해결하였다. 우선 책갈피에 의미있는 이름을 부여할 수 있는 과정을 간략화 했다. 또한 페이지를 북마크 하는 개념 대신에 박스를 이용하여 특정 컨텐츠를 기억할 수 있도록 했다. 이에 관련한 디테일은 추후 개발 과정에서 설명하도록 하겠다.

(2) 필기 어플리케이션: Samsung Note

Samsung note 는 PDF 를 불러와 이 위에 필기를 하는 것이 주 목표인 어플리케이션이라는 점에서 우리의 프로그램과 유사한 점이 많다고 생각했다. Samsung note 에서는 왼쪽에 페이지들의 축소판이 사진으로 보여지고, 책갈피 표시를 눌러 책갈피를 설정할 수 있다. 또한 책갈피가 된 페이지들을 모아서 볼 수 있다.

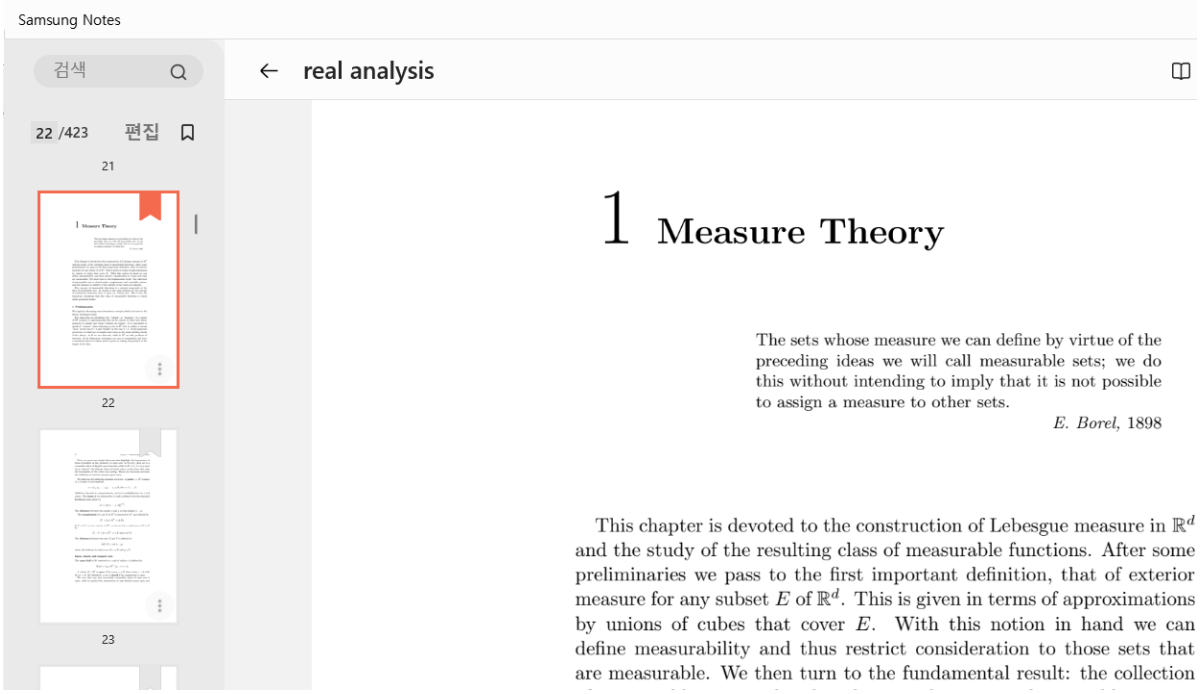


그림 2. Samsung Note 의 책갈피 사용 모습

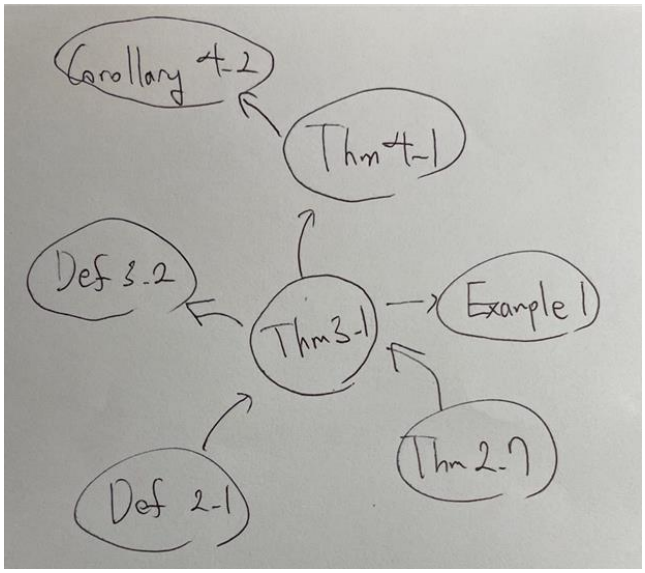
Samsung note 에서는 책갈피를 축소된 이미지로 보여주기 때문에 의미 있는 정보를 보다 쉽게 담을 수 있다. 하지만 그림이 없고 텍스트로 이루어진 문서의 경우, 책갈피가 페이지 전체의 이미지를 가지고 있으면 책갈피 사이의 구분이 어려워지게 된다.

3. 디자인 및 개발 과정

3.1 개발 아이디어의 흐름

이 주제를 선정한 이유는 앞에서 충분히 이야기한 것 같으니 넘어가도록 하자.

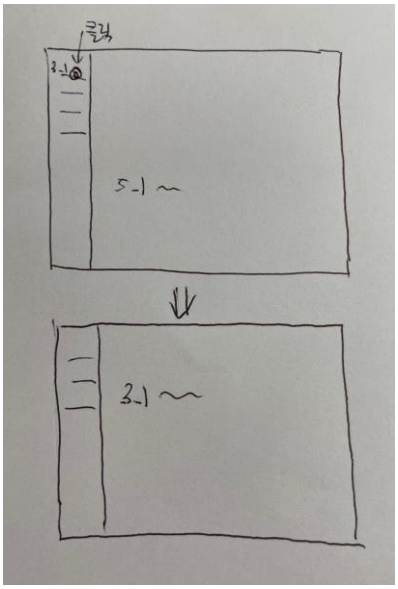
수학공부에 도움이 되는 툴을 만들기로 결심했을 때 처음 생각한 것은, scene scenario 다이어그램처럼 책을 이루는 여러 요소들을 평면상에 그려서 예쁘게 나타내주는 것이였다.



이 생각을 하게 된 것은 단순히 예쁘고 뿌듯할 것 같아서만은 아니고, 실제로 그래프를 그리는 방식으로 생각하는 사람들이 많기 때문이다. 어려운 수학을 배울수록 수학의 정리가 현실과 겹치는 부분이 적어지기 때문에, 그래프를 그리는 것은 수학 내용들을 추상화하기에 가장 범용적이고 간단한 방법이다.

하지만 이 방법에는 문제가 있었는데, 바로 콘텐츠들의 이름에 그 내용이 들어있지 않다는 점이다. 수학책에 적혀있는 내용이 한두가지가 아니기 때문에, 그것을 한눈에 나타내려면 각 컨텐츠들이 가질 수 있는 공간은 적어질 수 밖에 없다. 그래서 결국엔 콘텐츠의 이름을 적는 방법밖엔 없는데, 아주 유명한 정리가 아닌 이상, 보통의 콘텐츠에는 정해진 이름이 없고, 따라서 수학책은 숫자를 붙여서 이들을 정리해 놓는다. 하지만 Theorem 3.1 과 같은 이름을 보고 우리는 아무것도 떠오를 수 없기 때문에(그정도로 공부가 되어있지는 않은 상황을 상정하고 있다.), 우리는 이름만 적어서 만들어진 그래프는 별로 쓸모가 없다고 판단했다. 따라서 더 좋은 방법을 생각해 내기로 했다.

그래서 생각해 낸 것이 pdf 사이를 자유롭게 왔다갔다하는(caper 하는) 툴을 구현하는 것이다. 수학과가 아니더라도 긴 pdf 를 이동하느라 고생해본 경험이 있는 사람이 많을 것이기 때문에, 우리는 이 아이디어가 매우 좋다고 느껴졌다. 문제는 caper 을 할 곳을 지정해주거나, 컨텐츠 간의 관계를 지정해 주는 일을 사용자가 해야하고, 그 과정이 귀찮을 수 있기 때문에 우리는 그 방면에 집중하기로 했다.



초반에는 먼저 콘텐츠간의 관계를 표시하기 전에, 사용자가 지정해둔 곳을 빠르게 이동할 수 있는 기능을 먼저 구현하기로 했다. 그림과 같이 왼쪽 구석에 사용자가 지정해둔 곳을 페이지순으로 나열해 놓은 메뉴를 만들기로 했다. 하지만 앞서 말했지만 다른 pdf 뷰어가 그렇듯이 theorem 하나하나마다 위치를 저장하기 위해 이름을 입력해 넣어야 한다면 너무 불편할 것 같았다. 그래서 우리는 OCR 기능을 사용해서 이것을 해결하기로 했다.

OCR은 optical character recognition의 줄임말으로, 사진으로부터 글자를 읽어내는 기술이다. 이미 많은 곳에서 이것을 쓰고 있기 때문에 다행히도 우리가 직접 이를 구현할 필요는 없었고, tesseract 라는 라이브러리를 찾아서 쓰기로 했다. (tesseract 를 쓰니 mac 에서는 프로그램 작동이 안되던데, 조교님 컴퓨터에서도 그럴까봐 조금 걱정이 된다. Windows 에서는 정상적으로 작동한다.) 사용자가 콘텐츠의 이름을 selectionbox 로 선택하면, 두 개의 버튼이 만들어지는데, Hierarcy 버튼과 Content 버튼이다. Hierarchy 버튼은 왼쪽 끝에 나열되어 있는 목차 버튼으로, 다른 컨텐츠들과의 pdf 상에서의 위치 관계를 알려주기 때문에 Hierarcy 버튼이라고 이름 붙였다. Content 버튼은 Crop 을 한 콘텐츠 바로 위에 생기는 버튼으로, 콘텐츠 위치에 위치해 있기 때문에 content 버튼이라 부르기로 했다. Hierarchy 버튼은 자신과 함께 만들어진 Content button 을 기억하고 있어서, Hierarchy 버튼을 눌렀을 때 원하는 위치로 이동할 수 있다.

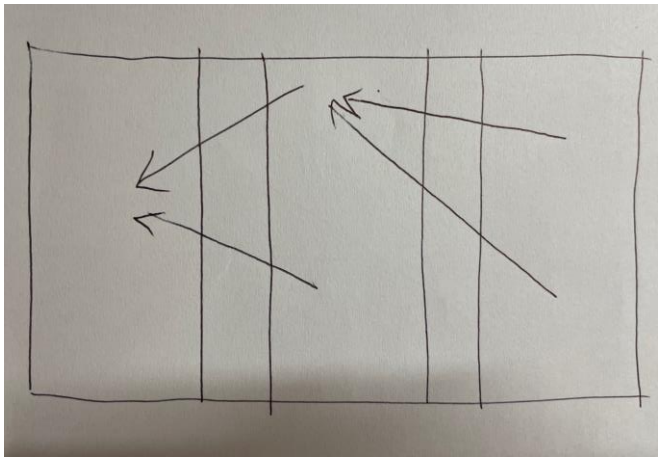
목차를 만든것만 해도 충분히 쓸만한 기능이지만, 목차를 만든것은 범용성이 높은 기능인 것 같고, 우리는 수학에 특화된 프로그램을 만들고 싶었기 때문에, 어떻게 하면 좀 더 발전시킬 수 있을지를 생각했다. 그렇게 생각해낸 것이 콘텐츠간의 관계를 지정해주는 것이다. 수학을 공부할 때, 예를 들어 theorem 4.1 에 대한 공부를 하고 있는데, 그것이 theorem 3.1 을 사용한다면, 우리는 theorem 3.1 에 대해 공부하고 싶어질 수 있다. 이 때 나중에 목차가 많아지면 theorem 3.1 을

찾기 귀찮아 지므로, pdf 상에서 theorem 3.1 을 연결해 주는, link button 을 만들어준다는 아이디어를 냈다. 자세한 인터렉션은 중간 경과 발표로 낸 영상에서 확인 가능하다.

Link button 은 Hierarchy 버튼을 pdf 쪽으로 끌어와서 만들 수 있는데, 이때 Hierarchy 버튼을 해당 콘텐츠를 이용하는 콘텐츠의 content button 위에 끌어와야 한다. (예를 들어, 위의 예시에서 theorem 3.1 의 Hierarchy 버튼을 theorem 4.1 의 Content 버튼에 끌어와야 한다.) 이것은 장단점이 확실하다. 먼저 장점은, 콘텐츠간의 관계를 쉽게 알아낼 수 있다는 점에 있다. Theorem 3.1 이 theorem 4.1 을 imply 하고, theorem 4.1 이 theorem 3.1 을 need 한다는 관계가 확실함으로, theorem 3.1 에도 자신이 theorem 4.1 을 imply 한다는 정보를 줄 수 있다. 이러면 theorem 3.1 의 content 버튼을 클릭했을 때, imply 버튼을 만들어 해당 theorem 이 무엇을 imply 하는지 보여준다거나 하는 기능을 넣을 수 있다. (아무 화면에다 넣으면, theorem 3.1 을 무슨 목적으로 화면에 나뒀는지 알 수 없다.) 하지만 단점은 이런 동작이 너무 자연스럽지 못하다는 점이다. Link 버튼을 원하는 곳에 놔두기 위해 해야하는 일이 많아지고, 또 무언가를 imply 하려는 것이 아니라 그냥 다시 보고싶어서 link 버튼을 만든 것일수도 있다.

그렇게 고민을 하던중에 교수님과 조교님이 좋은 아이디어를 주셨는데, 바로 같은 pdf 를 일렬로 여러개 정렬하는 것이다. (우리는 이때 좌우로 정렬된 pdf 들을 branch 라고 표현 하기로 했다.) 지금까지 콘텐츠들은 위아래로 배치되어 있었기 때문에 그들간의 관계를 선으로 이어서 표현하기가 까다로웠다. (Imply 버튼은 그렇게 해서 생각하게 된 것이다.) 하지만, 같은 pdf 를 여러개 연결한다면, 서로 관계가 있는 콘텐츠들을 좌우로 배치할 수 있기 때문에 굳이 Hierarchy 버튼을 쓰지 않고, content 버튼 사이를 이어서 관계를 지어줄 수 있다.

공부를 할 때 보통 어떤 콘텐츠를 보고, 해당 콘텐츠가 사용하는 콘텐츠를 보러가기 때문에, 우리는 branch 를 왼쪽에서 오른쪽으로 늘려나간다면, 공부를 할 때 왼쪽이 가장 상위의 (책으로 치면 가장 아래쪽의), 오른쪽이 가장 하위의 (가장 위쪽의) 콘텐츠가 자리잡게 될 것이라고 가정했다. 이를 바탕으로 어떤 branch 의 콘텐츠가, 해당 branch 좌측의 다른 branch 의 콘텐츠를 imply 한다면 화살표를 그리게 만들었다. 이렇게 하면 공부를 하던 중에 A 키를 눌러서 모든 content 버튼을 선택하면, 지금까지 branch 를 만들면서 넘어온 내용들이 어떤 관계를 갖는지 한눈에 알 수 있게된다!



굿이 한방향의 관계도만 가능하게 한 이유는 화살표를 아무지 잘 그려봤자, 화면을 축소하면 안보이기 때문에, 위에 적혀 있는 가정이 맞다면 한 방향만 그려져 있는 것이 오히려 이해에 좋기 때문이다.

3.2.1 코드 요소 구현 관련

3.2.1 PDF Rendering

(1) PDF Rendering

pdf 형식의 파일을 java 상에서 잘 다루기 위해서 PDFBox 라는 라이브러리를 사용했다. 이 라이브러리를 이용하기 위해 JPanel 을 extend 해서 CDPDFViewer 클래스를 만들었다. 이 클래스는 pdf 를 스크린 상에 렌더링 하기까지 필요한 요소들을 가지고 있다. 특히 문서 자체를 비롯해 Renderer, xform 등을 가지고 있다.

이를 위해 여러 메서드를 만들었다. 먼저 paintComponent 가 섬세하게 다루어졌다. 우선 전공서적의 경우 페이지 수가 많아 모든 페이지를 렌더링 하는 것은 비효율적이기 때문에 현재 xform 을 생각해서 스크린 안에 들어올 페이지만 렌더링 해서 그려주었다. 두번째로 나중에 구현될 caper 를 위해서 스크린 상의 포인트가 주어졌을 때 pdf 에 해당되는 페이지 및 페이지 상에서 x,y 위치정보를 받아올 수 있는 매서드가 구현되었다. 또 pdf 의 렌더링 단위는 페이지이고, 따라서 페이지 하나가 커지면 아무리 좋은 전력을 세워도 렌더링이 오래 걸릴 수밖에 없다. 따라서 Xform 에서 제한을 두어서 최대 확대 크기를 제한했다.

교수님과 조교님의 코멘트 이후에 같은 pdf 파일을 여러개 띄우는 방식으로 문서 사이를 caper 하는 방식으로 바뀌었고, 따라서 일렬로 여러 pdf 파일을 렌더링해야했다. 이런 방식은 일렬로

나열된 pdf 들의 위치(현재 보여지고 있는 페이지)를 따로따로 조절해야 때문에, panel 들이 공통적으로 가지고 있는 Xform 에 더하여 각 pdf 들의 world 좌표계 상의 위치를 저장했다.

(2) Scroll/Pan/Zoom

pdf 를 띄웠으면 페이지를 이동하는 것을 구현해야 한다. 먼저 가장 직관적으로 마우스 휠로 스크롤을 하면 페이지가 연속적으로 움직이도록 했다. 이때 마우스 포인터가 특정 페이지 위에 있으면, 해당하는 branch 가 움직이고, 그냥 빈 공간에 있으면 여러 branch 가 한번에 이동하게끔 만들었다. Ctrl 키를 누르고 드래그를 함으로써 드래그를 연속적으로 할 수 있도록 Pan 모드를 구현했다. Ctrl 을 누르고 마우스 휠을 스크롤하면 그 점을 중심으로 zoom 이 구현된다.

3.2.2 JSI 기반 그림 그리기 구현

(3) Drawing/Select/Color

그림을 그리고 선택하며 선택한 선들의 색과 굵기를 바꾸거나 지우는 기능을 기존 JSI 프로젝트를 바탕으로 옮겨왔다. 이때 pdf 렌더링이 오래 걸려서, event 를 받기위한 시간 간격이 많이 늘었는데, 이때로 부드러운 곡선을 그릴 수 있도록 수업 후에 따로 구현했던 Bezier curve 를 이용해 C1 연속성을 보장하는 곡선을 그리는 기능도 함께 가져왔다. 오른쪽에 Color 버튼을 만들어 3 가지 색과 하나의 하이라이터를 쓸 수 있도록 했다. 하이라이터는 굵기와 투명도를 따로 저장해서 가지고 있다. 수학을 할 때 다채로운 색을 사용해야 하는 경우는 없기 때문에, 복잡한 color chooser 보단 자유도가 떨어지지만 직관적인 방식을 채택했다.

3.2.3 레이어 구분

(4) Layer 구분

원래 CD 가 렌더링을 하는 과정은 JSI 와 같이 Canvas 에서 모든 작업을 해주는 것이었다. 하지만 우리 프로그램의 경우 pdf 와 ptCurve 가 동시에 존재하고, 나중에 버튼과 만든 버튼을 띄워줄 창이 별도로 필요했기 때문에 개념적으로 이를 다른 JPanel 의 확장된 클래스들로 구분했다. 지금은 PDFViewer, Canvas, ButtonViewer, SideViewer 이렇게 총 4 개의 패널이 있고, 이것들이 하나의 패널 안에 겹쳐서 보이는 구조이다. 렌더링해야하는 것이 많아진 만큼, 패널이 분리 되니 편한점이 많았다.

3.2.4 Crop 및 Hierarchy 버튼 구현

(5) Crop/OCR/Hierarchy

Crop 은 pdf 상에 표시된 개념을 목차로 표현할수 있게 만들어주기 위해 필요한 과정이다. 이렇게 만들어진 개념들은 hierarchy 에 저장되며, 이들 사이의 관계를 지정할 수 있다. c 키를 누르고 드래그를 하면 selection box 와 비슷하게 crop box 가 만들어진다. 앞서 PDFViewer 에서 만들었던 매서드를 바탕으로 crop box 를 정의하는 두 점을 이용해 페이지 상에 필요한 부분을 image 로 저장했다.

다음 단계로 crop 된 부분의 글씨를 인식하는 부분을 구현했다. image 를 따로 저장하지 않고 crop 된 부분의 buffered image 를 받아 tesseract 라는 library 를 이용해서 인식한 뒤에 해당되는 버튼을 만들고 hierarchy 창 위에 정렬해서 띄우도록 했다. Crop 된 부분은 푸른색 반투명한 content 버튼을 만들어 클릭했을 때 해당 개념이 사용된 부분을 나타내준다.

3.2.5

(6) Button 간의 linking/ Caper

중간고사 전의 버전에서는 xform 을 바꿈으로써 링크를 타고 페이지가 이동한 것처럼 보이도록 caper 를 구현했다. Caper 은 hierarchy 버튼이나 link button, imply 버튼을 통해 가능했다. Pdf 상에서 '뛰어다닌다'는 개념을 좀 더 부각하기 위해, 머물렀던 페이지들로 이동하는 기능을 구현했다. 이를 위해 history 의 관리가 중요했다. 히스토리는 결국 해당 시점의 xform 과 현재 history 상에 어느 위치에 있는지 index 를 가지고 있는 것이 중요하다. History 는 caper 할때만 저장되며, 마우스휠이나 PAN 을 통한 xform 의 변화는 저장하지 않는다. 키보드 좌우키를 이용해서 history 를 이동할 수 있고, caper 를 하게 되면 history 상에서 현재 위치에 해당하는 index 뒤의 기록을 지우고 caper 하기 직전의 xform 을 history 에 추가해준다.

Branch 개념을 사용하고 나서부터는 조금 변했다. 위아래가 아니라 좌우로 움직이게 되어있고, 순서대로 정렬되어있기 때문에 굳이 history 를 만들 필요가 없었다. 좌우키를 통해 좌우를 한칸(pdf 사이의 너비만큼)씩 움직일 수 있다. 이동을 위한 버튼을 누르면 focus 가 되어있는 branch 를 기준으로 오른쪽에 새로운 브랜치가 생기고, 원래 오른쪽에 있었던 branch 들은 사라진다. 지금까지와 관계가 없는, 조금 멀리 있는 곳을 공부하고 싶은 경우를 상정해서 (예를들어

시험범위가 3 단원 7 단원일 수 있다.) Side 에 있는 버튼을 오른쪽으로 클릭하면, 0 번째 브랜치의 해당 위치로 이동하게끔 만들었다.

4. 인터페이스 및 인터랙션

프로그램을 키면 터미널에서 어떤 pdf 를 열 것인지 묻는다. 원하는 pdf 파일의 경로를 입력할 수도 있고, 그냥 엔터를 치면 우리가 default 로 넣은 실해석학 책이 켜진다. 프로그램이 시작하면 패널이 화면 전체를 채우고, 책 표지를 볼 수 있다. 이전에 savefile 이 있다면, 그것도 함께 로딩된다.

Rotation 을 제외한 JSI 의 모든 기능이 구현되어있다. Ctrl 후 휠을 누르거나 드래그해서 Zoom 과 pan 이 가능하고, 펜을 움직여서 그림을 그릴 수 있으며, shift 이후 드래그를 통해 선택하고, 지울 수 있다. 이때 branch 를 만들어야 하기 때문에 panmark 나 selectionbox 는 하나의 branch 를 벗어날 수 없다. 여기에 추가로, mouse wheel 을 이용한 스크롤 기능이 추가되었다.

본격적으로 우리 프로그램만의 기능을 사용해보자. C 를 누른 상태에서 드래그해서 cropBox 를 만든 후, 마우스와 C 키를 순서대로 떼면 ContentButton 가 생기고, 사이드바에는 cropBox 안에 적혀있는 내용이 적혀있는 HierarchyButton 이 생긴다. Crop 을 여러번 하더라도, HierarchyButton 은 콘텐츠의 위치에 따라 정렬된다. 이제 스크롤을 통해 pdf 를 내려가다가 HierarchyButton 을 왼쪽 클릭하면, 새로운 branch 가 생기면서 HierachyButton 에 관련된 콘텐츠를 볼 수 있다. 다음번에 이 과정을 더 쉽게 하고싶다면, HierarchyButton 을 오른쪽으로 드래그 한 후 원하는 곳에 놓으면 HierarchyButton 과 같은 이름의 LinkButton 이 생긴다. 다음번엔 사이드바에서 HierarchyButton 을 찾지 않더라도 LinkButton 을 클릭함으로써 같은 효과를 볼 수 있다. LinkButton 을 드래그하면 위치를 옮길수도 있다. Branch 는 현재 focus 되어있는 페이지의 오른쪽에 생기고, 그 전에 오른편에 있었던 페이지들은 지워진다. HierarchyButton 중 하나를 마우스 우클릭하면, 0 번째 branch 의 해당 콘텐츠로 이동한다.

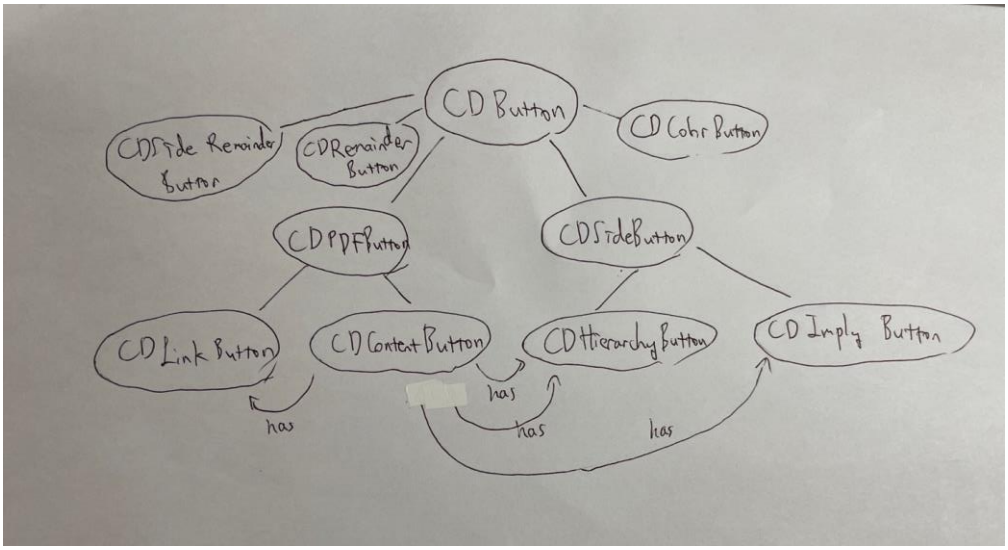
우측 ContentButton 과 좌측 ContentButton 을 마우스로 이으면, 우측이 좌측을 입증(imply)하는 상관관계가 생긴다. 우측 ContentButton 이 클릭되서 choosed 됬을 때에는 해당 콘텐츠가 무슨 콘텐츠를 입증할 수 있는지가 왼쪽 사이드바에 표시되고, 가까이 있을 때에는

화살표로도 표시된다. A 키를 누르면 모든 ContentButton 이 선택되고, 현재까지 공부한 내용들의 연결을 한눈에 볼 수 있다.

현재까지의 정리내용은 S 키를 통해 저장할 수 있다. 또 ContentButton 과 Link Button 은 선택 후 삭제할 수 있다.

5. 클래스 계층 구조 다이어그램

원하는 기능을 추가하다 보니 버튼이 너무 많아졌다. 그래서 버튼에 관련한 개념들을 정리할 필요성을 느꼈고, 그래서 버튼에 대한 Hierarchy 체계를 만들었다. 간단한 그림은 다음과 같다.



3 개의 abstract class 와 7 개의 concrete class 들로 이루어져 있다. 먼저 CDButton 은 최상위에 위치한 abstract class 로, 모든 버튼이 가질만한 특징들을 가지고 있다. Name, kind, ishighlighted 가 그것이다. (모든 버튼은 연관된 콘텐츠의 이름과, 자신의 종류를 가지고 있고, highlight 될 수 있다.) 이때 kind(종류)는 enum 의 형태로 나누어서 저장하는데, 이유는 나중에 서술하겠다.

CDPDFButton 은 PDF 위에 그려질 버튼들을 서술하기 위한 abstract class 로, PDF 상에서 버튼이 그리는 rectangle 을 가지고 있다. 반면 CDSideButton 은 사이드바에 그려질 버튼들을 위한 abstract class 이며, 버튼들이 어떻게 나열될 지는 정해져 있기 때문에, 자신이 가르치고 있는 콘텐츠의 y 좌표를 가진다.

여기 각각에 기능 구현을 위한 몇가지 기능을 추가하면 CDLinkButton, CDContentButton, CDHierarchyButton, CDImplyButton 의 4 가지 concrete class 를 만들 수 있다. 한 가지 주목할 점은 이들의 포함관계이다. CDContentButton 이 LinkButton, HierarchyButton, ImplyButton 을 모두 가지고 있고, 그 외에는 HierarchyButton 이 Content Button 을 가지고 있다. 콘텐츠 자체를 표시하고 있는 ContentButton 이 가장 중요하다는 생각에서, ContentButton 이 해당 콘텐츠와 연관된 모든 버튼들을 기억하게끔 해놓았다. 이는 실제로 말이 되는것이, 특정 내용에 대한 버튼이 만들어지기 시작하는 것이 Content 버튼을 만들고부터이기 때문이다. 직관적으로 생각해 보아도 link button 을 없앤다고 해도 ContentButton 은 없애야겠다는 생각이 들지 않지만, Content 버튼을 삭제하면 link button 등 관련된 버튼들을 같이 지우는 것이 맞다. 이런 컨셉은 save/load, 삭제 인터렉션 구현에 많은 도움이 되었다.

ColorButton 은 컬러 선택을 위한 버튼이다. ColorButton 까지 하면 5 개의 concrete class 가 만들어지고, 나머지 두개의 버튼은 사이드바 전체, 나머지 부분 전체를 가지는 CDSideRemainderButton 과 CDRemainderButton 이다. 이 두 코드는 왜 필요한가 의문이 들 수 있는데, 이것은 scene scenario diagram 과 코딩을 깔끔하게 할 수 있도록 해주는 구조다.

JSI 를 만들 때에는 화면의 어느부분을 클릭하던지 해야하는 인터렉션이 같았다. 하지만 버튼이 많은 지금은, 화면을 클릭할 때 마다 5 개의 버튼과, 사이드바 나머지 부분, pdf 뷰어 나머지 부분의 7 가지 경우의 수를 고려해야한다. (심지어 이것들의 위치가 계속 변한다..!) 하지만 이것을 scene 내에서 if-else 문으로 나타내는 것은 매우 헷갈리고 어려울 것이므로, 우리는 이것을 keyEvent 를 다룰 때 처럼 case 문으로 나타내면 좋겠다는 아이디어를 내게 되었다. 최상위 class 인 CDButton 을 따라 7 개 각각의 버튼은 각각의 kind 를 가진다. 그리고 ButtonManager 은 마우스 포인터를 받고, 클릭된 버튼을 output 으로 뱉는다. 그러면 우리는 ouput 으로 나온 버튼의 kind 를 읽어 switch 문에 이용할 수 있다. 이것을 이용해서 우리는 버튼이 많더라도 나름의 체계를 유지하며 아래와 같이 깔끔하게 코딩할 수 있었다.

```
@Override
public void handleMousePress(MouseEvent e) {
    CD cd = (CD) this.mScenario.getApp();
    CDButton button = cd.getButtonMgr().checkButton(e.getPoint());
    CDButton.Button kind = button.getKind();
    switch (kind) {
        case CONTENT:
            cd.getSideViewer().setImplyMode(
                cd.getButtonMgr().getCurMouseContentButton());
            CDWorldButtonScenario.getSingleton().
                setCurHandlingContentButton(
                cd.getButtonMgr().getCurMouseContentButton());
            button.setHighlight(true);
            CDWorldButtonScenario.getSingleton().setStartPoint(
                e.getPoint());
            XCmdToChangeScene.execute(cd, CDWorldButtonScenario.
                ContentPressedScene.getSingleton(), this);
            break;
        case HIERARCHY:
            CDSideButtonScenario.getSingleton().
                setCurHandlingSideButton(
```

6. 씬/시나리오 다이어그램

이번 장에서는 Caper Doc 의 씬/ 시나리오 다이어그램에 관해 설명하겠다. 우선 프로그램이 복잡한 관계로 모든 내용을 한장에 넣는 것 보다는 (1)전체 씬/ 시나리오 사이의 이동 관계를 나타내는 다이어 그램과 (2)각 시나리오 별로 세부적인 내용을 정리한 다이어 그램으로 나누어 정리했다. 따라서 Caper Doc 의 전체적인 흐름을 보고 싶다면 (1)번 다이어그램을, Caper Doc 의 구체적인 기능들을 하나하나 살펴보고 싶다면 (2)번 다이어그램들을 확인하면 된다.

Scene Scenarion diagram 에 앞서, 아래 그림에서는 Scene/ Scenario diagram 에서 사용된 약자 및 Key Event/ Mouse Event/ Mouse Motion Event 의 표기법을 정의한다.

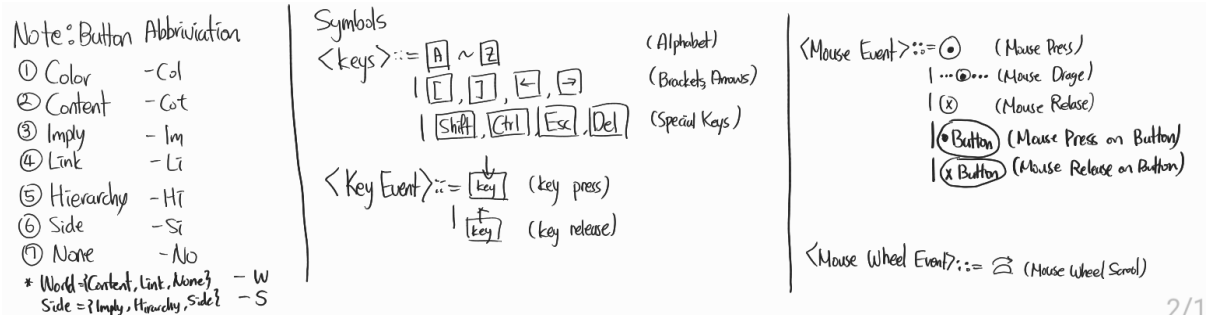


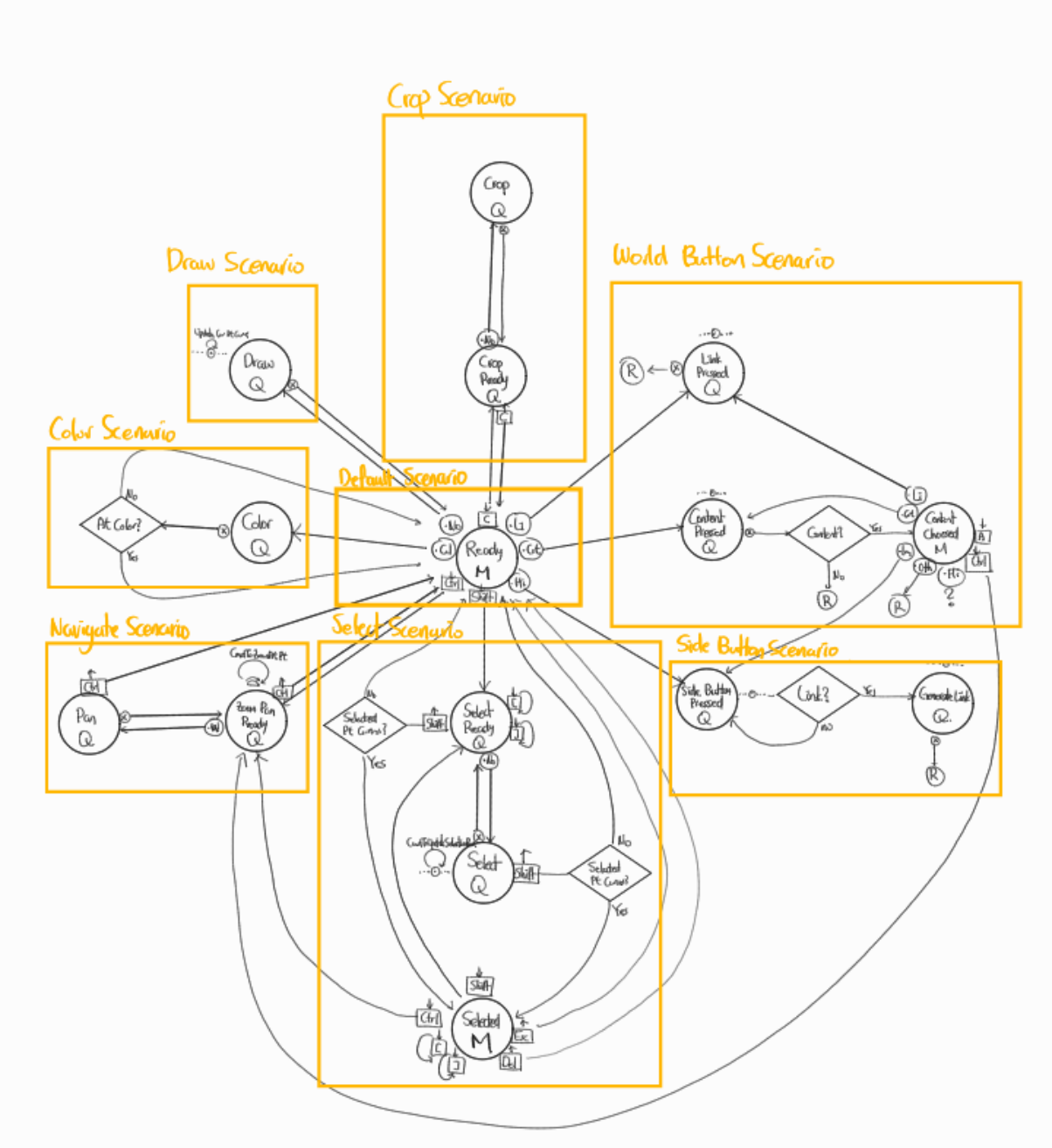
Diagram 에서 유의할 만한 점은 Button 은 키보드와 다르게 화면 위에 있는 객체로, 특정 버튼 위에서 마우스가 Press/ Release 되는 Event 및 특정 버튼 위에서 마우스 휠이 Scroll 되는 Event 를 각각 기호로 나타냈다. 또한, 버튼의 종류가 크게 World 버튼과 Side 버튼으로 나뉘게 되는데, World 버튼에 속하는 Content, Link, None 을 통틀어 W 로 나타내고 Side 버튼에

속하는 Imply, Hierarchy, Side 를 통틀어 S 로 나타냈다. 이 외에도 각 7 가지 버튼은 약자를 사용하여 Scene/Scenario diagram 에 표기했다.

1. 전체 Scene Scenario Diagram

Caper Doc 은 총 8 개의 Scenario 로 구성되어있다. Scenario 는 (1) 가장 기초가 되며 다른 Scenario 사이를 이동하기 위한 중심이 되는 Default scenario, (2) PDF 위에 그림을 그리기 위한 Draw Scenario, (3) 그림을 그리는 펜의 종류를 선택할 수 있는 Color Scenario, (4) 그린 곡선들을 선택할 및 편집할 수 있는 Select Scenario, (5) 화면들을 이동 및 스크롤 할 수 있는 Navigate Scenario, (6) PDF 의 특정 부분은 잘라 Content 버튼을 만들어주는 Crop Scenario, (7) Hierarchy 창에 띄워진 버튼을 world 위에 끌어 링크를 만들기 위한 Side Button Scenario, (8) World 위에 생성된 버튼들을 움직이고 새로운 창을 띄우기 위한 World Button Scenario 로 구성되어있다.

각 Scenario 들은 여러 개의 Scene 들로 구성되어 있는데 이 Scene 간의 이동은 아래 그림을 통해 확인 할 수 있다. 다음 그림은 전체 Scene Scenario Diagram 이다.



2. 각 Scenario 별 세부 Diagram

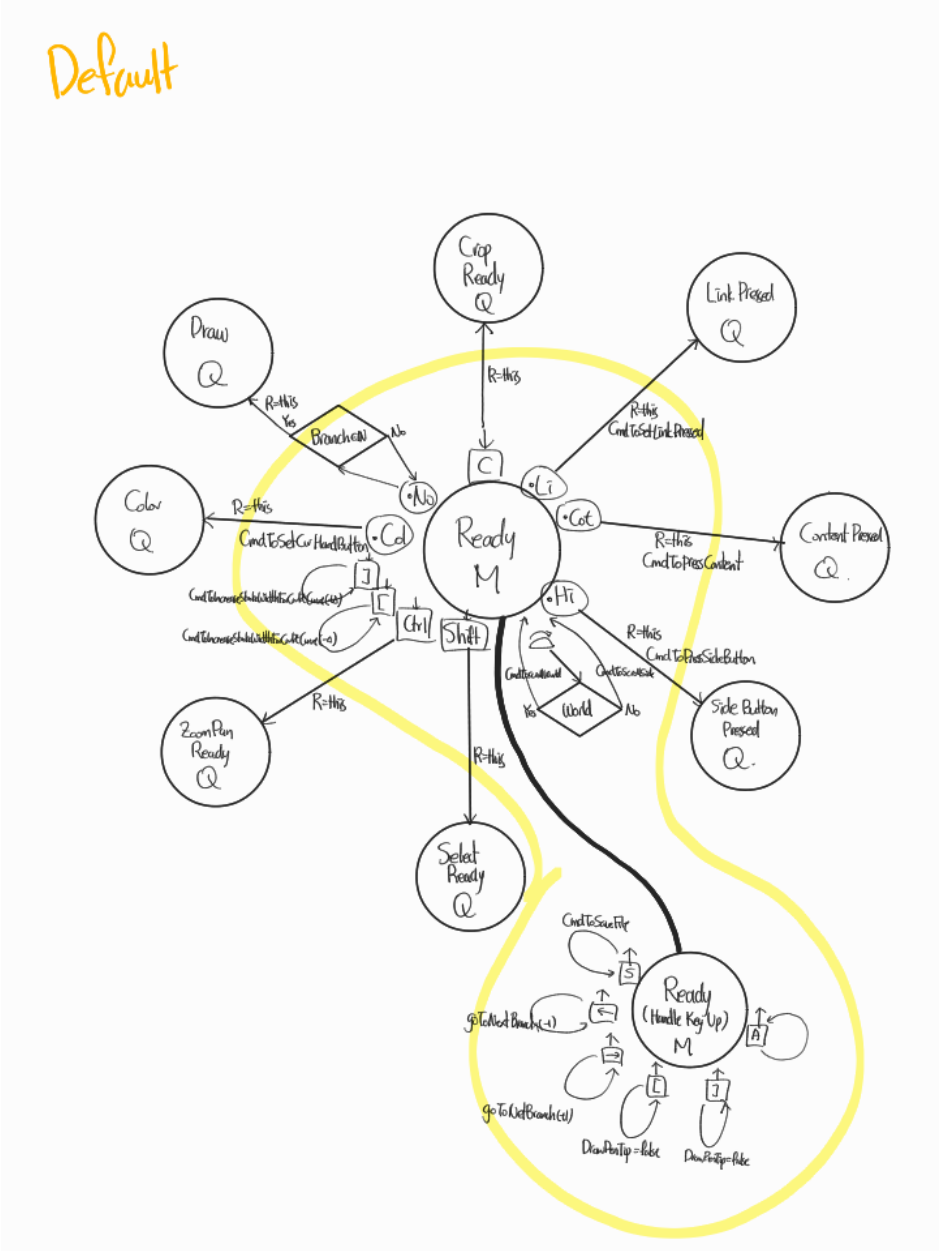
각 시나리오의 세부 다이어그램은 해당 시나리오에 속해있는 Scene 들과, 그 Scene 들에서 부터 바로 접근할 수 있는 다른 Scenario 의 Scene 들까지 표기를 했다. 하지만 다른 Scene B 에서 부터 바로 현재 대상이 되는 Scenario 에 속해있는 Scene A 로 접근 할 수 있다고 해도 Scene B 는 표기하지 않았다. 즉 이 다이어그램은 현재 Scenario 에서 어디로 뻗어 나갈 수 있는지만을 이야기 해주고, 어느 상황에서 이 Scenario 로 돌아올 수 있는지는 이야기 해주지 않는다.

후자의 내용은 앞서 표기된 '1. 전체 Scene Scenario Diagram'에서 확인 가능하다. 현재

대상이 되는 Scenario 는 하이라이트로 다른 Scenario 의 Scene 들로 부터 구분되었다. 이제 Scenario 들을 하나씩 살펴보자.

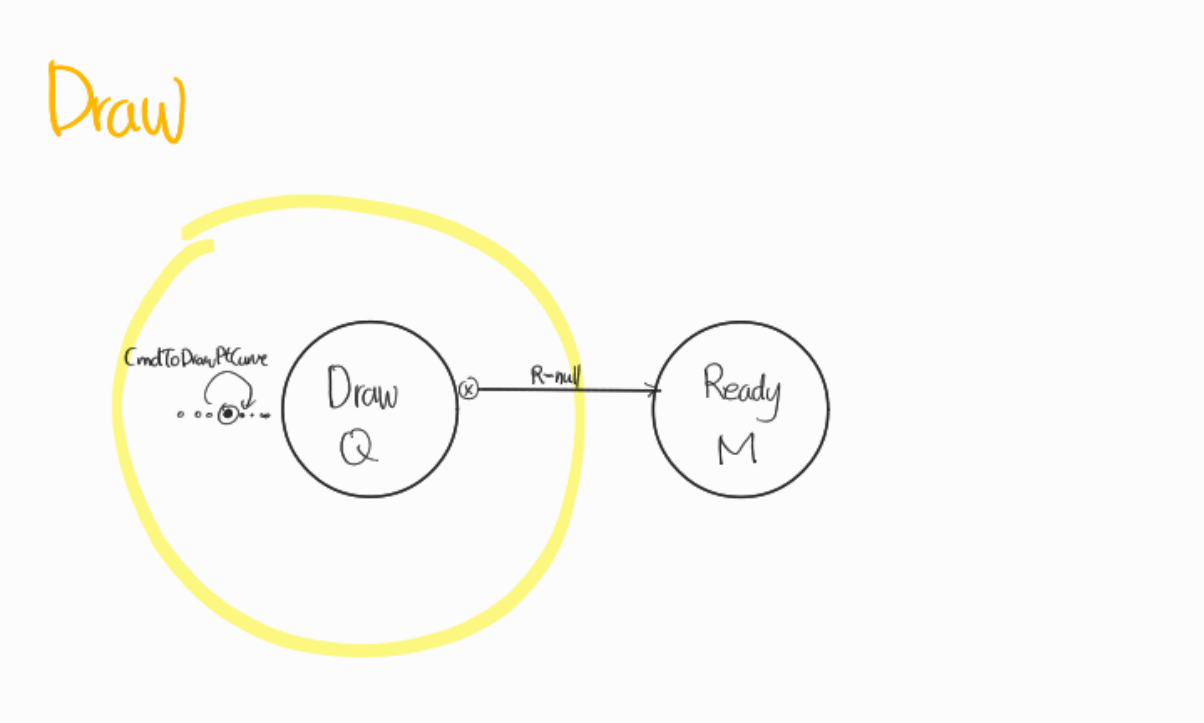
(1) Default Scenario

Default Scenario 에서는 화면 속 다양한 버튼과 키보드 입력을 통해 다른 Scene/ Scenario 로 가기 위한 중심점이다. Default Scenario 는 Ready Scene 하나로 구성되어 있으며, Ready Scene 에서 화면에 보여질 수 있는 버튼들에 대해, press 되었을 때 어느 Scene 으로 뻗어 나가야 하는지를 보여준다. Handle Key Up 을 처리해야 하는 부분은 공간의 부족으로 우측 하단에 따로 빼서 표기했다.



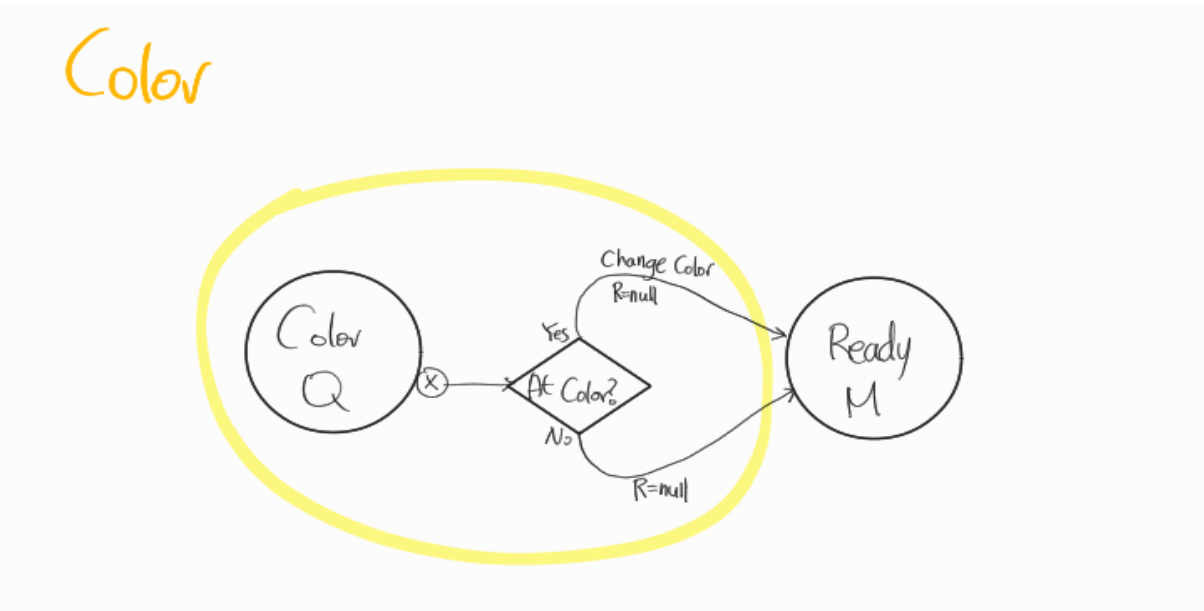
(2) Draw Scenario

그림을 그리기 위한 Scenario 로 Draw Scene 하나만을 가지고 있다. Draw Scene 은 Drag 될 때 마다 현재 가지고 있는 Pt Curve 를 업데이트 해준다.



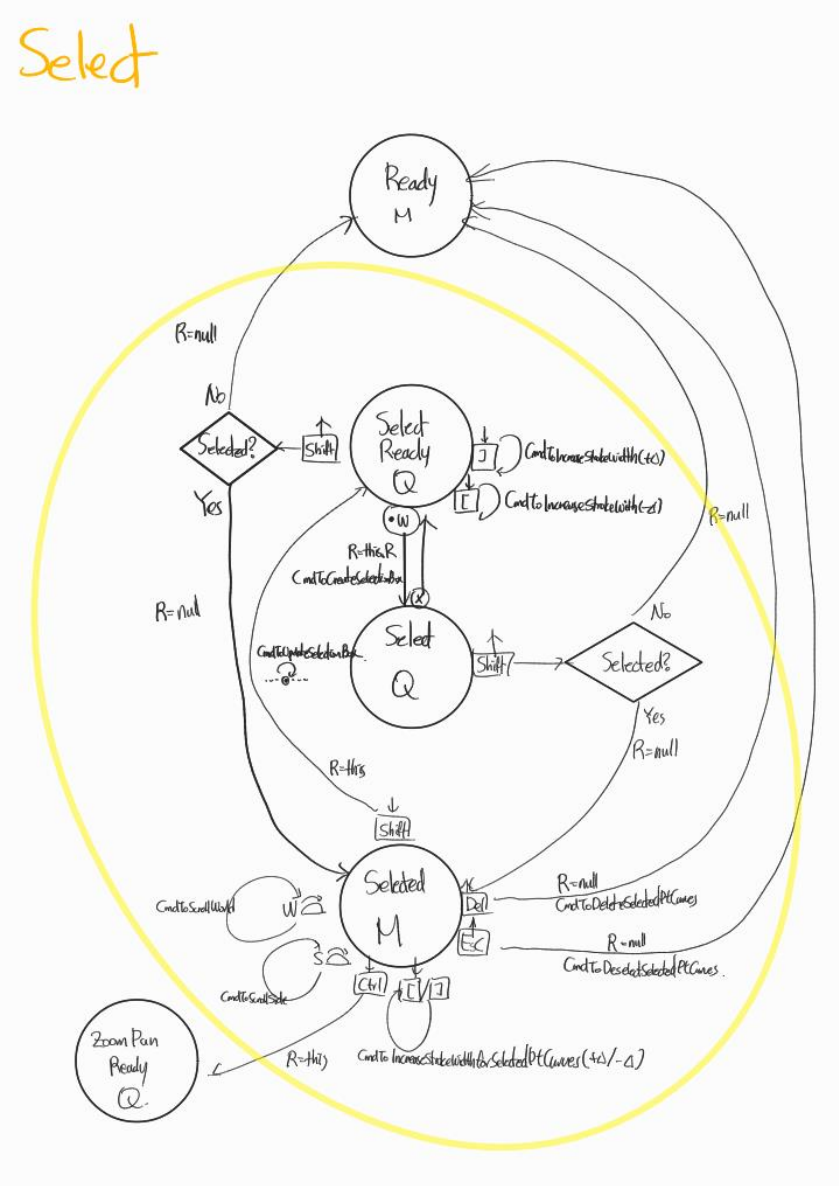
(3) Color Scenario

Color Scenario 역시 Color Scene 하나만을 가지고 있다. Color Scene 은 Default Scenario 의 Ready Scene 에서 Color Button 을 누름으로써 들어오게 되는데, 펜이 Release 될 때에도 역시 Color Button 위에서 떼어졌는지를 판별하여 색을 바꿀지 여부를 결정한다.



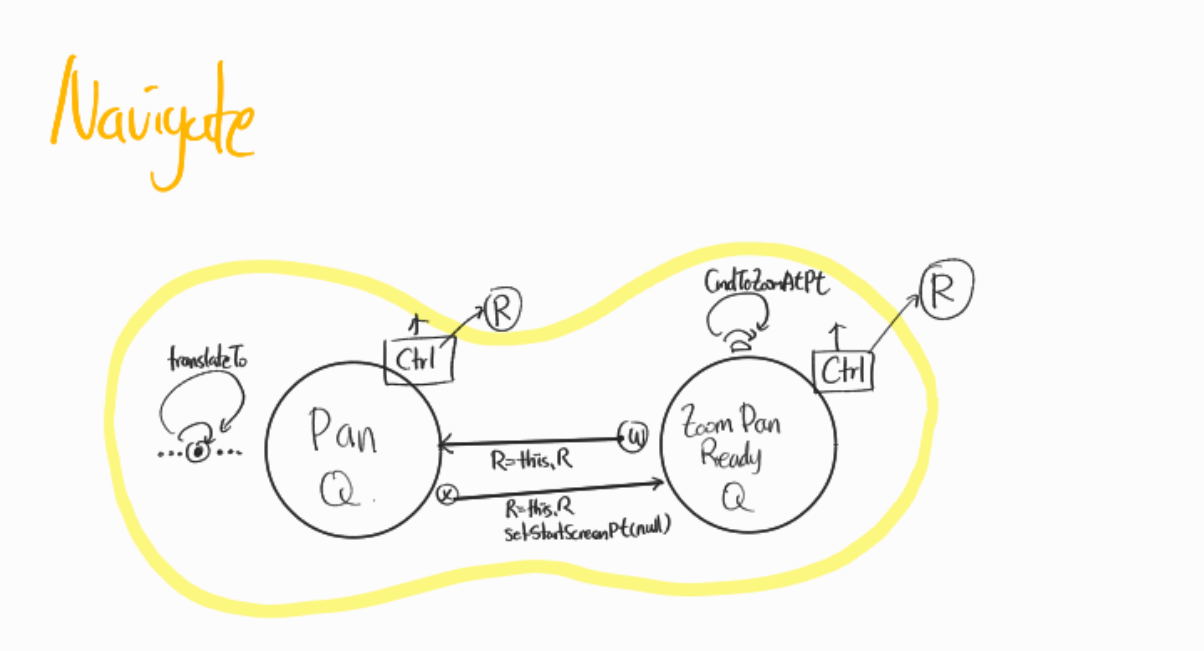
(4) Select Scenario

Select Scenario 는 Select Ready, Select, Selected 로 이루어진 3 개의 Scene 들로 구성되어 있다. Default Scenario 의 Ready Scene 이나 Select Scenario 의 Selected Scene 에서 Shift 키를 누르면 Select Ready 로 들어오게 된다. 마우스를 누르게 되면 Selection box 가 생성되고 Drag 를 할 때 마다 Selection box 가 업데이트된다. 마우스를 떼면 Select Ready 로 돌아오고, shift 키를 때었다면 Selected 된 PtCurve 가 있는지 여부에 따라 Selected 혹은 Ready Scene 으로 돌아가게 된다. 이 외에도 Selected Scene 에서는 선택된 PtCurve 들에 대해서 굵기를 바꾸는 작업을 해줄 수 있도록 설계됐다. 또한 Default Scenario 의 Ready Scene 에서와 마찬가지로 마우스 휠을 Scroll 할 때, 마우스 포인터의 위치가 pdf 위에 있는지 여부에 따라서 마우스 포인터가 있는 pdf 만을 스크롤 하거나 화면 전체를 스크롤 할지를 결정했다.



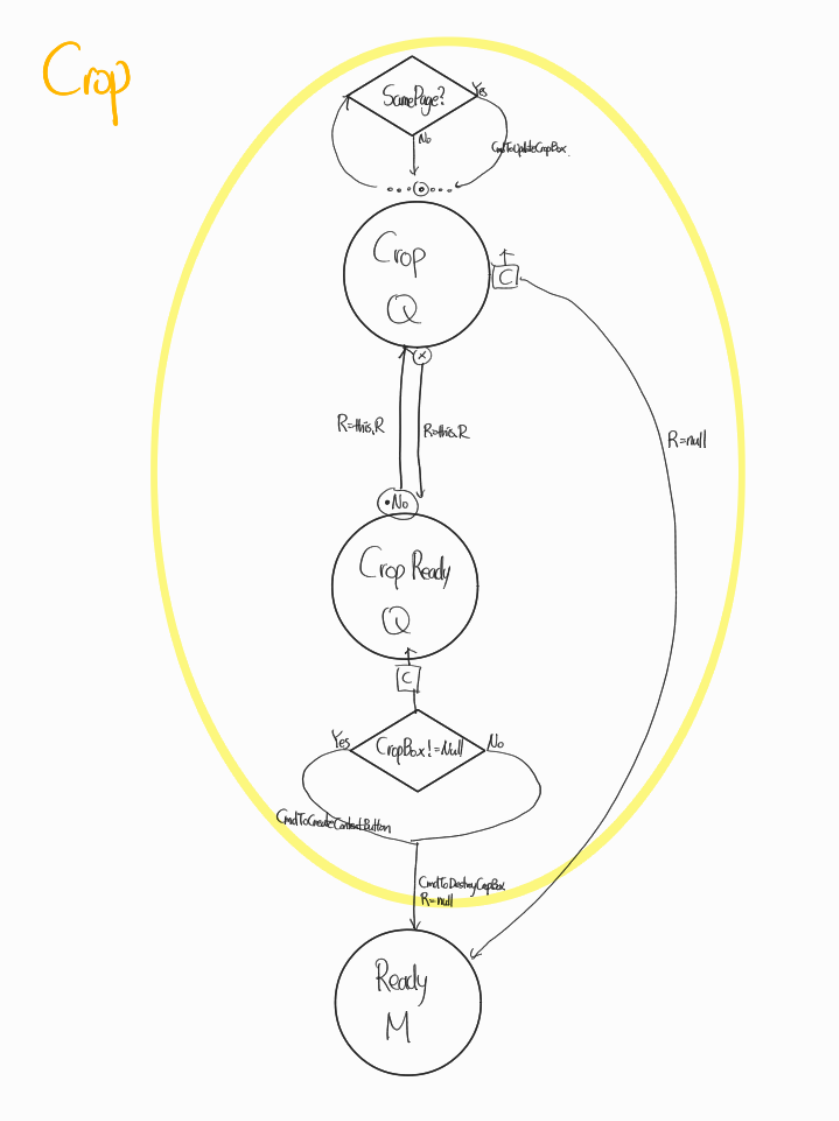
(5) Navigate Scenario

Navigate Scenario 는 ZoomPanReady 와 Pan 라는 두 Scene 으로 구분되어 있다. 이 Scenario 는 Default Scenario 의 Ready Scene 이나 Select Scenario 의 Selected Scenario 에서 접근이 가능하다. 마우스를 클릭하면 Pan 으로 이동하면서 마우스가 드래그 되는 동안 화면이 따라 움직이고, 마우스를 놓게되면 다시 Zoom Pan Ready 로 돌아오게 된다. Zoom Pan 에서 마우스 휠을 움직이면 화면 전체가 zoom 이 된다. Pan Scene 에 비해 Zoom 관한 Scene 을 따로 만들지 않은 이유는 마우스 휠은 돌리는 시작과 끝이 명확하지 않기 때문이다.



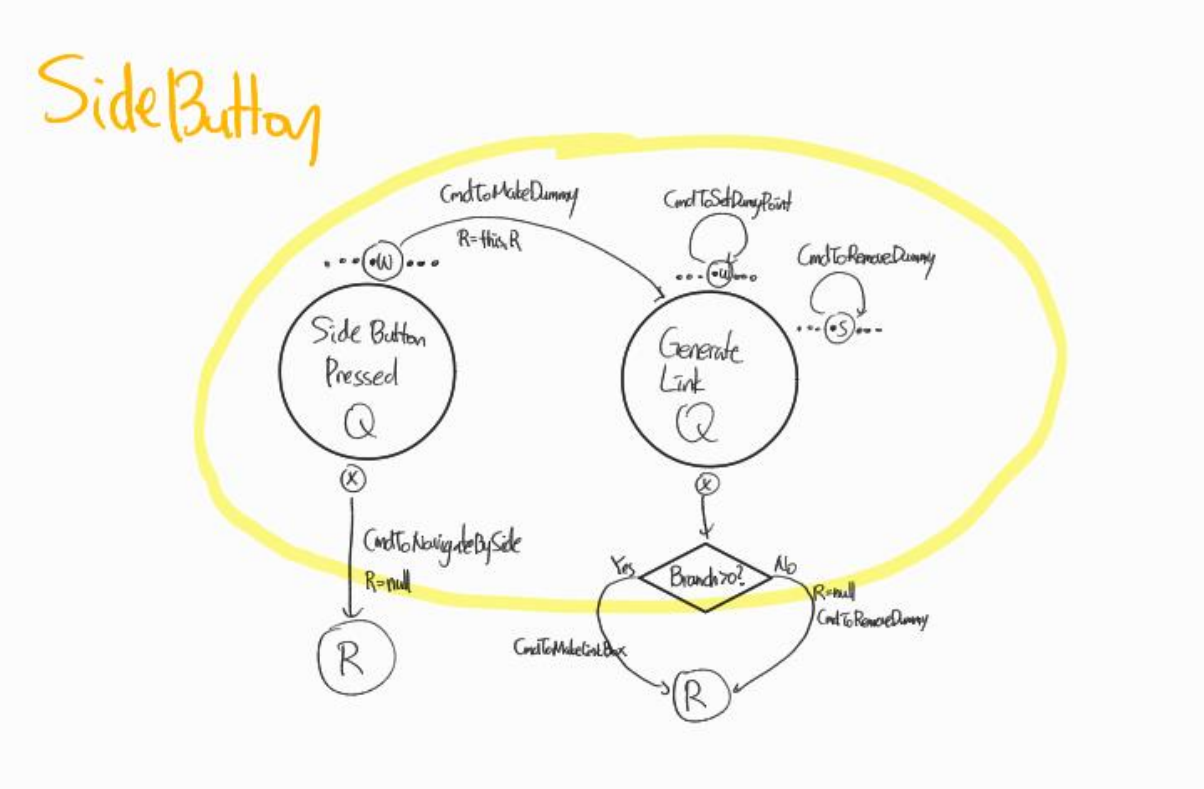
(6) Crop Scenario

Crop Scenario 는 Select Scenario 와 비슷한 구조로 짜여있다. 한편 Scene 은 Crop Ready, Crop 으로 구성되어 있는데, Select Scenario 의 Selected Scene 에 대응되는 Cropped Scene 이 없다. 그 이유는 Crop box 는 만들었을 때 화면에 있는 내용을 무조건 인식해서 Hierarchy 를 만들어 주기 때문에 Selection box 처럼 안에 무언가가 있는지 여부를 판별할 필요가 없기 때문이다.



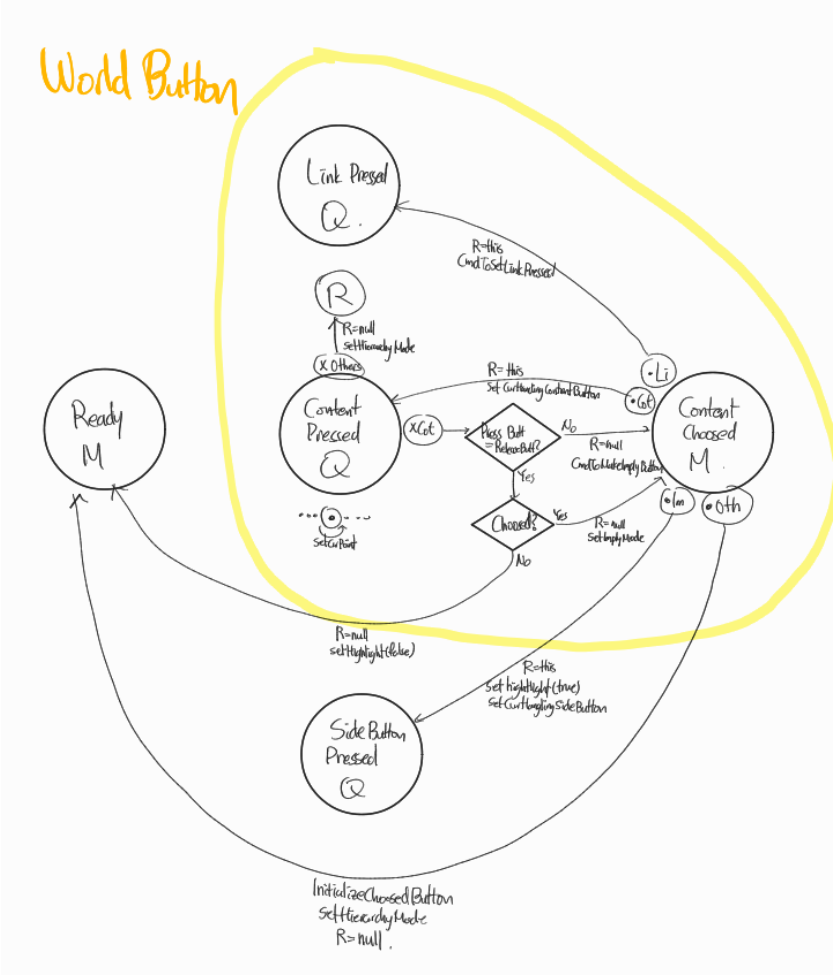
(7) Side Button Scenario

Side button Scenario 는 Side button 을 이용한 작업을 처리하는 Scenario 로, side 에 있던 hierarchy button 을 가져와서 link 버튼을 생성하는 등의 일을 할 수 있다. 이 Scenario 에는 Side button Pressed 와 Generate link 라는 두개의 Scene 들이 있다. Hierarchy 창에 있는 버튼을 누르게 되면 Side button pressed scene 에 진입 할 수 있게 된다. 만약이 버튼을 누른채로 hierarchy 창 안에만 머물러 있으면 아무 일이 생기지 않지만, 만약 그 버튼을 hierarchy 창 밖으로 끄집어 낸다면 link generate scene 으로 이동하게 된다. 여기서 링크가 실수로 만들어 졌을 수도 있기 때문에 버튼을 떼지 않은채로 side 로 다시 hierarchy 로 이동하게 되면 버튼을 없애준다. 한편 World 창 상에서 계속 드래그를 하고 있는 중이라면 dummy button 이 만들어져 실제 버튼을 끌고 있는 것과 같은 애니메이션을 보여준다.



(8) World Button Scenario

World button scenario 는 world 에 위치해 있는 버튼으로 할 일들을 처리해준다. 이 Scenario 는 Content Pressed, Content choosed, link pressed 라는 3 개의 Scene 들로 구성되어 있다. 특히 Content 가 눌려진 Content pressed scene 의 경우 다양한 실행할 수 있는 선택지가 다양하며, 각 상황에 맞는 일을 처리해준 것을 볼 수 있다. Content pressed scene 에서 *oth 라는 기호는 *other 의 약자로 Scene 에 이미 표기된 버튼을 제외한 나머지 버튼을 눌렀을 때 라는 의미이다.



7. 결론 및 레슨

<요약>

수학공부를 위한 프로그램을 만들었고, 완성도 있는 결과물을 얻을 수 있었다. 우리가 만든 프로그램은 전자기기로 pdf 를 볼 때 위아래의 일차원적인 움직임으로 돌아다니는 데에서 벗어나, 각종 버튼과 2 차원적인 배치로 pdf 를 '뛰어다니는'듯이 움직일 수 있다. 또한 사용자는 책의 내용의 핵심 콘텐츠를 자신만의 목차로 정리할 수 있고, 콘텐츠간의 관계를 정해줄 수 있으며, 내용을 습득하는 중에 그런 관계를 한눈에 볼 수 있다.

<소감>

이렇게 큰 프로그램을 처음부터 끝까지 우리의 힘으로 만들어 본 것이 처음이라 신기한 경험을 한 것 같다. 처음 Caper Doc 을 기획하고 개발을 하기까지 팀원들 사이에 정말 많은 피드백이 오갔고, 구현에 대한 아이디어를 위해 토론도 정말 많이 했다. 이렇게 큰 프로젝트가 오류 없이 잘 작동 할 수 있는 것은 수업시간에 프로그램의 틀을 짜는 방법을 배웠던 덕분인 것 같다. 프로그램의 골자가 되었던 X Package 역시 Caper Doc 개발에 큰 도움이 되었다.

Caper Doc 은 대학생인 우리들의 입장에서 실제 쓸 가치가 있는 프로그램을 만든 프로젝트였다는 점에서 더욱 큰 의미로 가다오는 것 같다. 공부를 하면서 느꼈던 여러 어려움을 구체화하고, 이를 해결하는 프로그램을 직접 설계한 점이 프로젝트를 제작하는 큰 동기가 되었다. 프로젝트를 개발하면서 계속 새로운 아이디어가 떠올라 이를 서로 공유했고, 우리에게 주어진 시간 안에 얼마 만큼의 구현을 해낼 수 있는지를 가늠하는 것도 큰 도전이었다.

Scene/ Scenario diagram 을 통해 프로젝트의 전체적인 개요를 파악하고 각 Scenario 별로 Scene 사이의 diagram 을 다시 그려보면서 엄밀한 부분들을 채워 나갔다. 특히 각 Scene 마다 사용자가 처해 있는 상황을 생각해 보니, 프로그램을 어떻게 디자인 해야 할 지 더욱 감이 잡혔던 것 같다. 또한, 개발에 진행한 우리 둘 뿐만 아니라 제 3 자인 친구를 데리고 와서 프로그램을 써보게 하기도 하면서 꽤나 튼튼하고 견고한(?) 우리의 프로그램을 보고 기뻐하기도 했다.

프로젝트를 진행하면서 아쉬웠던 점들도 있다. 먼저 막연하게 아이디어를 자꾸 떠올리다 보니까 프로젝트에 대한 욕심이 계속 생겼다. 이런 아이디어를 가지고 있음에도 시간을 더 투자하지 못해서 구현하지 못한 부분에 대해서는 계속 아쉬움이 남는 것 같다. 그래도 이 프로젝트가 단순히 이번학기에 끝나는 것이 아니라 교수님께서 늘 말씀하셨듯 계속 이끌어나갈 수 있는 프로젝트니까 시간을 내서 짬짬히 구현해보고 싶었던 것들을 구현해보고 싶다.

피드 포워딩에 관해서도 아쉬운 점이 몇가지 있었다. 처음 이 프로그램을 사용해보는 사용자라면 단축키나 마우스를 통한 명령어를 가쳐 주어야 하는데, 이에 관련된 메뉴얼을 만들거나, 비주얼적으로 더 잘 피드 포워딩 해줄 수 있었던 부분들이 있는데 이 부분이 아쉬운 것 같다. 당장 하나 예를 들자면 링크 박스 생성할 때 어느 위치에서 생성이 가능하고 어느 위치에서는 불가능 한지를 버튼의 색으로 피드 포워딩 해주었다면 더 좋았을 것 같다.

이러한 점들을 제쳐 두고라도 우리는 단기간에 꽤나 의미있는 일을 한 것 같다. 이 프로그램이 완성되어 가면서 점점 쓸만해져 갔고, 이러한 과정을 직접 경험하면서 스스로 많이 성장한 것 같다. 기한이 정해져 있는 일들은 주어진 시간 안에 보여줄 수 있는 최선의 결과를 보여주는 것이 이상적이다. 하지만 앞으로도 프로젝트를 하더라도 자기 자신과 주변 사람들을 잘 챙길 수 있는 사람이 되면 좋겠다.

<여담>

프로젝트를 하던 중 둘이 함께 신학관의 롯데리아에 가서 햄버거를 먹었었다. 그 전에는 아무 생각이 없었는데, 롯데리아의 키오스크가 어떤 인터렉션에도 안정적으로 움직이는 것을 보며 오늘도 일용할 햄버거를 먹을 수 있는 것에 대한 감동을 느꼈다. 수업시간에 교수님께서 자동차 디자인을 하다보면 지나가던 자동차가 어떻게 디자인되었는지 보인다고 하셨듯이, 이제 우리도 주변의 소프트웨어들이 어떤 씬 시나리오 다이어그램을 바탕으로 만들어졌을지 생각해보기 시작하는 단계인 것 같다. 프로젝트를 통해 이만큼 실력을 기를 수 있었던 것은 정말 좋은 기회였고, 그런 기회를 주신 교수님과 조교님께 다시한번 감사의 말씀을 전합니다.

8. 참고 문헌