디지털콘텐츠프로세싱 기말과제 보고서

디지털미디어디자인학과 201582023 이정우

* 목적

웹사이트 상에서 구동되는 저용량이되 화려하고 인터렉션이 가능한 javascript기반 언어로 짜여진 배경을 제작.

* 작동방법

p5.js로 제작하고 나니 스케치에서 로딩되는 파일이 있으면 로컬에서 CORS문제로 인해 로딩이 되지않고, 서버에 호스팅해야 한다는 것을 안게 되었습니다. 따라서 웹 호스팅을 한 뒤에 실행하여야 하여 교수님이 보실 수 있도록 미리 2군데에 호스팅 하였습니다.

<https://zao95.github.io/Project-Digital-Contents-Processing/>

<http://222.100.111.249/processing_Final/>

두 군데 모두 연결이 되지 않거나 컴퓨터 사양 관계로 정상적인 작동이 되지 않을 경우에는 index.html과 같은 공간에 있는 시연영상으로 채점해주시기 바랍니다.

* 제작 과정

참조한 예제[[1]](#footnote-1)는 processing[[2]](#footnote-2) 언어로 작성되어 있으며, processing.js[[3]](#footnote-3)도 지원 가능하도록 되어 있습니다. 그에 따라 processing언어와 processing.js를 이용하여 사이트배경으로서 제작하려 하였지만, 몇 가지 문제에 부딪쳐 p5.js\*로 변경하여 제작하게 되었습니다. 문제점은 아래와 같습니다.  
1. processing.js\*에서 processing의 pixel edit 부분을 제대로 계산하지 못합니다.  
2. processing.js의 처리속도가 java -> js의 변환 과정을 거치는 탓에 느려집니다.

아쉽게도 processing.js에서 p5.js 언어로 변경하는 프로그램이 없었기 때문에, java 강좌 사이트[[4]](#footnote-4)와 javascript MDN 사이트[[5]](#footnote-5)를 참조하여 직접 java에서 javascript 언어로 변경하였습니다.

또한 예제는 PGraphic을 이용하여 text를 그린 뒤, 픽셀을 검사하여 가깝고 불투명한 픽셀에 파티클이 다가가는 형태로 작성되어 있으나, 본 과제는 변수에 createGraphic을 통해 공간을 열고 그 안에 이미지를 넣은 뒤, 픽셀을 검사하는 형태로 되어있습니다.

그리고 사이즈를 유동적으로 변경할 수 있도록 수정하였습니다. 예제같은경우는 가로 700px, 세로 300px로 고정되어 있지만, 해당 과제에서는 javascript로 window의 스크롤과 기타 border 등을 제외한 window inner size를 구하여 canvas의 크기가 유동적으로 변경될 수 있도록 하였습니다. 또한 창 크기는 언제든 변경될 수 있기 때문에, 창 크기가 변경될 시에는 그것을 이벤트로 하여 프로그램이 재시작되도록 하였습니다.

구체적인 프로그램의 순서는 html에서 p5 라이브러리를 가져온 뒤, 윈도우 사이즈 등을 위해 사용되는 jqeury 라이브러리를 가져오고, p5언어로 짜여진 javascript 파일을 가져옵니다. 그러면 p5로 짜여진 javascript를 p5 라이브러리가 해석하여 html에 canvas 태그 형태로 그려주게 됩니다. 이 사이에 jquery로 윈도우 사이즈를 측정하여 canvas의 크기를 정해주게 되는데, canvas로 그린 뒤에 canvas의 크기를 변경하면 픽셀로 이루어진 canvas를 강제로 확대하는 꼴이 되어 프로그램의 이미지가 모두 깨져버리기 때문입니다.

파티클과 관련된 부분은 먼저 createGraphic속성을 가진 변수를 만들어 해당 변수에 이미지를 불러줍니다. 이후 이 변수를 loadPixels()을 이용하여 픽셀단위로서 리스트형태로 불러줍니다. 픽셀은 java기반의 processing에서는 color 자료형으로 나와, 픽셀과 리스트 숫자가 같지만, javascript기반의 p5에서는 red, green, blue, alpha 순서로 4개의 변수형태로 나눠지기 때문에 pg.pixels[coordIndex \* 4 + 3] != 0 라는 조건으로 투명하지 않은 픽셀만 파티클들이 가도록 프로그래밍했습니다.

실행 시, 캔버스 위에 띄운 테스트 div는 pointer-events를 명시적으로 보여드리기 위함입니다. 실제 클릭 시, 테스트 div에는 클릭이 되지 않고, 그 뒤에 출력되고있는 canvas에 영향이 가고 있음을 알 수 있으실 것입니다. 실제 웹사이트 제작 시에는 버튼이나 스크롤 가능해야하는 부분 등, 일부 요소를 제외한 다른 모든 요소에 pointer-events: none;을 넣어 canvas의 최대한 넓은 영역이 마우스를 이용한 인터렉션이 가능하게끔 작동 시키는 것이 이상적일 것 같습니다.

* 배운 점 & 느낀 점

저는 프로세싱을 배우면서 새로운 가능성을 보았습니다. 처음에는 processing을 이용해서 웹사이트의 배경을 만들면 용량이 적고 인터렉션이 가능하여 획기적이겠다! 라는 생각을 하였지만, 이윽고 processing이 웹사이트에 삽입되려면 processing.js라는 라이브러리로 변환되는 과정을 거쳐야하기 때문에 너무나 긴 로딩 시간으로 사용자 경험에 치명적인 악영향을 줄 수 있다는 점에 실망하였습니다.

하지만 p5라는 javascript 기반 processing언어의 발견으로 다시금 가능성을 엿보았고, 그것과 css의 pointer-events= none; 이라는 명령이 합쳐지면, 웹사이트에서 z-index가 낮게 배경으로 깔리더라도 인터렉션이 가능해지며, 용량도 적고 로딩도 빠른 좋은 배경이 될 것이라 생각합니다. 다만 용량 대신 클라이언트의 사양에 많은 부담을 주게 되지만, 현재 성능이 급격하게 상승하여 상향표준화된 모바일 하드웨어 덕분에 크게 문제는 없을 것이라 생각합니다. 실제 테스트 결과 최신기종은 아닌 노트8에서도 700x700이라는 해상도에서 particle을 이용해 490,000개의 픽셀과 수백개의 파티클이 프레임마다 움직이는 애니메이션을 구동해보았으나, PC구동을 상정한 애니메이션에도 불구하고 프레임은 10~20으로 안정적이였습니다. 만약 모바일에서 원활하게 돌아가게끔 다시 손보아 가로 픽셀을 줄이고, 애니메이션을 간략화시킨다면 프레임이 최소 30정도는 유지될 것이라 예상됩니다.

상기한 점을 보아 차후 웹디자인을 할 때에 사용하면 많은 도움이 될 것이라 생각됩니다. 물론 이런 비주류적인 목적이 아닌 차후 취직 후의 기업 박람회에 참가할 기회가 된다면 인터렉션이 가능한 전시 면에서도 큰 도움이 될 것이라 생각합니다. 픽셀단위 이미지분석과 수정 등 또한 현재 배우고 있는 머신러닝과 관련된 부분이 많아 정말 많은 도움이 되었습니다. 이번 한 학기동안의 경험이 향후에도 많은 도움이 되어 저의 큰 자산이 될 것 같습니다. 한 학기동안 감사했습니다.

참고 사이트

1. <https://www.openprocessing.org/sketch/377231>
2. <https://programmers.co.kr/learn/courses/17>
3. <https://developer.mozilla.org/ko/docs/Web/JavaScript>
4. http://p5js.org

1. 참고 사이트 1) <https://www.openprocessing.org/sketch/377231> [↑](#footnote-ref-1)
2. p5.js : processing의 javascript용 언어이다. p5.js라이브러리를 먼저 로드하고 p5.js 스크립트를 로드하는 형태로 간단하게 사용이 가능하고, 별도의 컨버트가 필요하지 않아, 초기로딩속도가 빠르다. 다만 processing과는 세세하게 많은 차이가 있다. [↑](#footnote-ref-2)
3. processing.js : processing언어로 짜여진 .pde파일을 웹사이트에서 사용하기 위한 컨버팅 라이브러리. [↑](#footnote-ref-3)
4. 참고 사이트 2) <https://programmers.co.kr/learn/courses/17> [↑](#footnote-ref-4)
5. 참고 사이트 3) <https://developer.mozilla.org/ko/docs/Web/JavaScript> [↑](#footnote-ref-5)