HTML5 Canvas 기반 오픈소스 이미지 에디터 라이브러리 개발*

정종윤 ○ 박성배

경희대학교 컴퓨터공학과

wormwlrm@khu.ac.kr sbpark71@khu.ac.kr

Opensource Image Editor Library Development based on HTML5 Canvas

Jongyoon Jeong○ Seong-Bae Park

Department of Computer Science and Engineering, Kyung Hee University

요 약

HTML5부터 등장한 Canvas API를 이용하면 웹에서도 이미지 편집을 비롯한 다양한 그래픽 기술을 처리할 수 있다. 하지만 기본적으로 제공하는 API의 추상화 단계가 낮고, 그래픽 도메인에 대한 지식을 별도로 학습해야 한다는 단점이 있다. 따라서 대부분의 경우에는 Canvas API를 직접 활용하기보다는 완성된 라이브러리를 활용하는 방법을 선택한다. 본 논문에서는 이미지 편집에 초점을 맞추어, 2D 그래픽 기반의 Canvas API를 활용한 오픈소스 이미지 에디터를 구현한다. 이미지 에디터에서 제공하는 다양한 편집 기능들을 어떠한 Canvas API와 디자인 패턴으로 구현 가능한지를 연구하고, 이 과정에서 마주치는 한계점들을 극복하는 방안에 대해 알아본다.

1. 서 론

HTML5부터 등장한 Canvas API는 그래픽을 표현하기 위한 웹 표준 API다. 이는 HTML5 이전까지는 웹에서 그래픽을 표현하기 위한 표준이 정해져 있지 않았다는 것을 의미한다. 초창기 웹은 단순한 문서의 역할에 가까웠기 때문에, 웹에서 기본적으로 제공하는 기능만으로는 풍부한 그래픽을 표현하기가 어려웠다. 따라서 Canvas API가 등장하기 전까지 웹에서는 SVG(Scalable Vector Graphic) 혹은 Adobe사의 Flash를 대체제로 이용하곤 했다[1]. 그 사이 웹은 인터넷의 발달, 모바일 스마트 디바이스의 대중화, 그리고 뛰어난 접근성을 바탕으로 애플리케이션 플랫폼으로 진화하게 되었다. 이에 HTML에서도 별도의 플러그인 없이 사용자 상호작용이 가능한 멀티미디어 지원에 대한 필요성이 제기되었고, HTML5부터는 그래픽을 전문적으로 표현하는 Canvas API가 표준으로 지정되었다.

이후 Canvas API가 상용화되면서 웹에서도 다양한 그래픽 기술을 처리할 수 있게 되었다. Canvas API는 JavaScript를 이용한 로우 레벨 그래픽 라이브러리인 WebGL을 호환하기 때문에, 2D 뿐만 아니라 3D 그래픽 역시 제작이 가능하다. 이를 이용해 애니메이션, 데이터 시각화, 이미지 편집, 실시간 비디오 처리 등 복잡한 기능을 별도의 플러그인 없이 구현 가능하다. 또한 웹 표준 API이기 때문에 특정 운영체제나 플랫폼에 구애받지 않는 독립적인 애플리케이션의 개발이 가능하다는 장점이 있다.

하지만 Canvas API는 기본적으로 제공하는 API의 추상화 단계가 낮고, 그래픽 도메인에 대한 지식을 별도로 학습해야 한다는 단점이 있다. 따라서 개발과 유지보수에 드는 비용을 절약하기 위해서는 일반적으로 필요한 기능을 제공하는 라이브러리를 쓰거나 혹은 관련 연구를 바탕으로 개발을 진행하게 된다.

기존에 Canvas API를 이용하여 진행된 선행 연구들을 살펴보면 다음과 같다. 장석우는 Canvas API를 이용한 플랫폼 독립적인 게임을 구현하는 연구를 수행했다[2]. 박영수는 Canvas API를 이용해 2D 이미지를 3D 입체 이미지로 변환하는 방법을 제안하는 연구를 수행했다[3]. 마지막으로 권미라는 Canvas API를 활용하여 사용자와의 동적 상호작용을 지원하는 웹 기반의 시각적 공간 분석 환경을 설계하고 구현하였다[4]. 이처럼 웹이 애플리케이션 플랫폼으로서 꾸준히 성장하고 있고, 표현하고자 하는 미디어의 종류 역시 다양해지고 있기 때문에 Canvas API에 대한 연구는 앞으로도 지속될 것이다.

본 논문에서는 이미지 편집에 초점을 맞추어, Canvas API를 활용하여 2D 그래픽 기반의 오픈소스 이미지 에디터를 구현하고자 한다. 1장에서는 본 연구를

* 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학 사업의 연구결과로 수행되었음(2017-0-00093)

수행하게 된 전체적인 개요와 배경에 대해 설명하였다. 2장에서는 이미지 에디터 개발에 필요한 관련 연구에 대해 기술할 것이다. 3장에서는 다양한 이미지 편집 기능에 사용되는 기술들을 어떤 Canvas API와 디자인 패턴을 이용해 구현 가능한지 알아보며, 기능 구현에 대한 한계점 역시 알아본다. 마지막으로, 4장에서는 본 논문의 결론 및 향후 연구방향을 제시한다.

2. 설계 고려사항

2.1 추상화

Canvas API는 기본적으로 제공하는 추상화 단계가 낮다. 이는 캔버스에 렌더되는 컴포넌트들을 객체지향적으로 관리하기가 어렵다는 것을 의미한다. 이미지 에디터는 단순히 하나의 이미지를 회전하고 잘라내는 것뿐만 아니라 도형이나 텍스트의 생성과 속성 조절 기능이 포함될 수 있으므로 컴포넌트 간에도 다양한 이벤트가 발생한다. 하지만 객체지향적인 구조 없이 모든 컴포넌트들을 관리하기는 쉽지 않다.

따라서 본 논문에서는 2D 그래픽에 대한 Canvas API를 JavaScript 기반으로 추상화하여 사용할 수 있게 해주는 Konva.js 라이브러리를 사용하고자 한다. Konva는 Canvas API의 2D Context를 확장하여 추상화된 객체를 제공할 뿐만 아니라 객체 편집 및 중첩 구조 관리, 브라우저 호환, 성능 최적화 기능 등을 추가로 제공하기 때문에 Canvas API를 고수준에서 활용하기에 적합하다고 판단했다.

2.2 상태 관리

바닐라 JavaScript에서 Konva를 활용하는 것도 가능하지만, 여전히 데이터와 뷰(View)가 불일치한다는 문제점이 있다. 따라서 UI 라이브러리로는 React를 사용하고자 한다. React는 선언형 뷰를 지원하기 때문에 뷰가 어떻게 그려질 것인지를 명시하면 데이터에 따라 뷰가 알아서 갱신된다. 또한 반복되는 코드를 컴포넌트 단위로 설계하여 재사용할 수 있기 때문에 코드 관리 관점에서도 유용하다.

2.3 패키지 형태로 릴리스

이번 연구는 라이브러리를 제작하는 프로젝트이다. 따라서 프로젝트의 결과물을 JavaScript 개발 환경에서 패키지 의존성으로 설치할 수 있어야 한다. 일반적으로 JavaScript 생태계에서는 NPM을 패키지 매니저로 사용하기 때문에, NPM에 이번 프로젝트를 릴리스 할 예정이다.

릴리스를 위해서는 여러 모듈로 관리되고 있는 JavaScript들을 단일 파일로 묶어주는 번들링(bundling) 과정이 필요하다. 이 때 JavaScript 표준 모듈 시스템으로 번들링을 가능하게 해주는 Rollup.js를 사용할 예정이다. 이를 ES6 모듈이라 부르는데, 이를 이용하면 정적 코드 분석을 통해 사용되지 않는 코드를 제거하는 트리 쉐이킹(Tree shaking)이 가능해진다. 따라서 라이브러리의 경량화를 함께 챙길 수 있다는 장점이 있다.

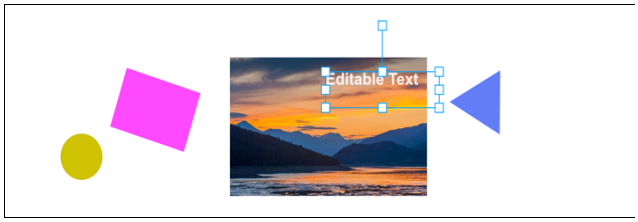
3. 기능 구현

3.1 이미지와 도형 생성, 편집 (Create Shapes)

사용자는 임의의 스타일을 가진 이미지와 도형을 캔버스에 배치시킬 수 있어야 한다. 뿐만 아니라 크기 조절, 색상 변경, 드래그 앤 드롭(Drag and drop), 도형 간 z-index 조절 등의 기능이 가능해야 한다.

이는 Circle, Rectangle, Triangle 등의 도형 컴포넌트를 사전에 먼저 정의하고, 각 도형에 Transform 기능을 추가한 Transformable Polygon 컴포넌트를 제작함으로써 구현이 가능하다.

3.3 포커싱 (Focusing)



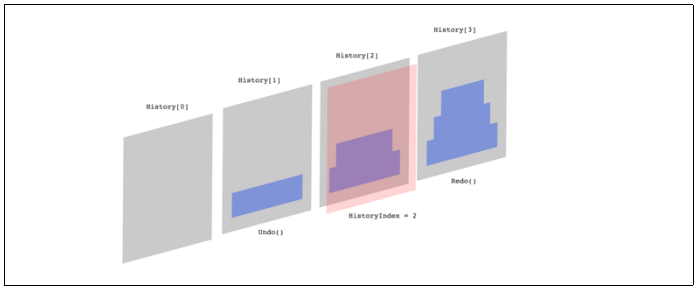
[그림 1] 여러 컴포넌트 사이에서 포커스 된 텍스트 컴포넌트

사용자는 일반적으로 하나의 컴포넌트를 선택하고 상호작용한다. 사용자는 선택한 컴포넌트에 대한 시각적인 피드백이 필요하며, 시스템은 사용자의 상호작용 결과를 선택한 컴포넌트에 적용시켜야 한다. 따라서 시스템은 현재 선택된 컴포넌트를 인지할 필요가 있다. 따라서 각 컴포넌트 별로 고유한 ID를 부여하고, React의 상태를 이용해 전역적으로 선택된 컴포넌트 ID값을 관리한다. 선택된 컴포넌트는 Transformer를 보여줌으로써 시각적인 피드백을 준다. 선택된 컴포넌트는 변환 기능이 활성화되며, 캔버스의 빈 영역을 클릭하거나 ESC 키를 누르는 경우 등에는 포커스를 해제한다.

3.4 실행 취소와 다시 실행(Undo and redo)

실행 취소와 다시 실행 기능은 스냅샷 기반의 메멘토(Memento) 디자인 패턴을 이용해 구현할 수 있다. 모든 상태 변경 스냅샷을 관리하는 Caretaker는 React의 전역 상태를 이용해 구현한다. 또한 상태를

저장하고 불러오는 Originator 역할은 캔버스가 담당한다. 그리고 Caretaker에서는 현재 몇 번째 상태를 가리키고 있는지를 저장하는 인덱스가 필요하다.



[그림 2] 스냅샷을 활용한 히스토리 아키텍처

매번 편집 이벤트가 발생할 때마다 Caretaker에 현재 스냅샷을 추가할 수 있다. 만약 실행 취소나 다시 실행 이벤트가 발생했을 때에는 스냅샷을 가리키는 인덱스를 1 더하거나 빼면서 조절한다. 이 때 캔버스는 현재 가리키는 인덱스의 스냅샷에 해당하는 데이터를 렌더하여야 한다. 또한 실행 취소 상태에서 새 편집 이벤트가 발생한 경우에는 해당 인덱스 이후의 스냅샷을 모두 삭제하는 방식으로 작업 관계 구조의 일관성을 유지할 수 있다.

3.3 드로잉 (Drawing)

드로잉은 HTML의 onmousedown 이벤트를 리스닝하여, 매 이벤트가 발생할 때마다 해당 시점의 x, y 좌표를 이산적인(Discrete) 좌표 배열로 저장하고 이를 렌더하는 Line 컴포넌트를 구현함으로써 만들 수 있다.

3.3 수정 가능한 텍스트(Editable text)

[그림 3] 텍스트 내용을 수정할 때 textarea로 대치한 모습

HTML 스펙에 따르면 캔버스는 기본적으로 캔버스 내에 인풋(Input) 요소로 텍스트를 수정하는 기능을 제공하지 않는다. 따라서 사용자의 텍스트 입력을 받아야 하는 경우에는 직접 텍스트 입력 기능을 구현하여야 한다.

이는 Text 컴포넌트의 x, y 좌표와 width, height를 계산하고 해당 위치에 textarea 엘리먼트를 생성하는 방식으로 해결 가능하다. 이 외에도 contenteditable 속성이 활성화된 HTML 요소를 이용할 수도 있다. 다만 두 방법 모두 외부 요소에서 발생한 이벤트를 캔버스 내부로 직접 전달해야 한다.

3.5 저장 및 복구 (Save and restore)

편집한 데이터의 저장 및 복구에 있어서는 Konva의 직렬화(Serialization)와 역직렬화(Deserialization) 기능을 사용한다. 기본적으로 이미지 라이브러리에서는 캔버스에 그려진 컴포넌트들을 JavaScript의 객체(Object) 타입으로 저장하며, 이들을 배열(Array) 형태로 관리하기 때문에 이를 JSON 타입의 파일로 직렬화하거나 JSON 파일을 읽어와 편집 상태를 복구하는 역직렬화 기능 구현이 가능하다.

하지만 JavaScript에서 JSON은 원시 타입(Primitive Type)의 데이터와 중첩 객체만을 저장할 수 있기 때문에, 이미지 객체나 적용된 필터 메서드 등의 경우에는 이를 직렬화할 수 없다. 따라서 이를 구현하기 위해서는 현재 상태를 직렬화 데이터에 담을 수 있는 데이터 타입으로 변환하는 전처리 과정과, 직렬화된 데이터를 실제 컴포넌트 그래픽에 반영하는 후처리 작업이 필요하다. 구체적인 방법으로는 Base64 인코딩을 이용하거나 기법이나 약속된 문자열을 사용할 수 있다.

4. 결 론

본 논문에서는 HTML5의 Canvas API를 활용하여 이미지 에디터 라이브러리를 제작하고, 각 기능들의 구현 방법과 한계점에 대해 연구하고 소스 코드를 공개하였다(https://github.com/wormwlrm/react-konva-image-editor). 고수준의 추상화를 제공하는 라이브러리를 활용하고, 상황에 맞는 디자인 패턴을 사용함으로서 이미지 에디터의 기능을 구현할 수 있었다. Canvas API의 범위를 넘어서는 요구사항에 대해서는 기술적으로 우회하면서도 사용자 경험을 해치지 않았다는 점에서 의의가 있다.

참고문헌

[1] Vitaly Friedman, Smashing Book 6: New. Frontiers In Web Design, 2018
[2] 장석우, 허문행, HTML5 캔버스를 이용한 플랫폼 독립적인 게임의 구현, 한국정보통신학회논문지, 제18권, 제12호, pp. 3042-3048, 2014
[3] 박영수, HTML5에서 직선의 기울기를 이용한 2D to 3D 입체 이미지 변환, 디지털융복합연구, 제12권, 제12호, pp. 521-528, 2014
[4] 박미라, 박기호, 안재성, HTML5 Canvas를 활용한 시각적 공간분석 환경의 설계와 구현, 한국지리정보학회지, 제14권, 제4호, pp.44-53, 2011