[표지]

2020 정보통신 2학기 프로젝트
작품 설명서

작품명	Javascript 웹 게임
한 줄 요약 설명	HTML5, CSS3, Javascript로 개발한 비주얼 노벨 웹 게임입니다.

작 품 명	Javascript 웹 게임
팀원 1 (팀장)	10607 김동희
팀원 2	10608 김일중
팀원 3	10610 남현종

[본문]

1. 작품 개요

가. 개발 동기 및 기대효과

(콘텐츠 선정 사유, 개발동기, 작품의 유용성 및 운영방향, 기대효과 등 기재)

사람들이 이 게임을 플레이 하는 것으로 큰 즐거움을 얻을 수 있었으면 해서 기획하고 개발하였다.

나. 제작과정

	날짜 또는 기간	제작과정
개발일정	~1/24	스토리 제작
	1/25	엔진 로직 디자인, 엔진 구현 일부 개발, 게임 페이지 HTML 작성
	1/26	엔진 로직 디자인 완성, 엔진 구현 개발 완성, 게임 페이지 CSS 디자인, 엔진에 이벤트 레지스터 등록, 메인 페이지 개발

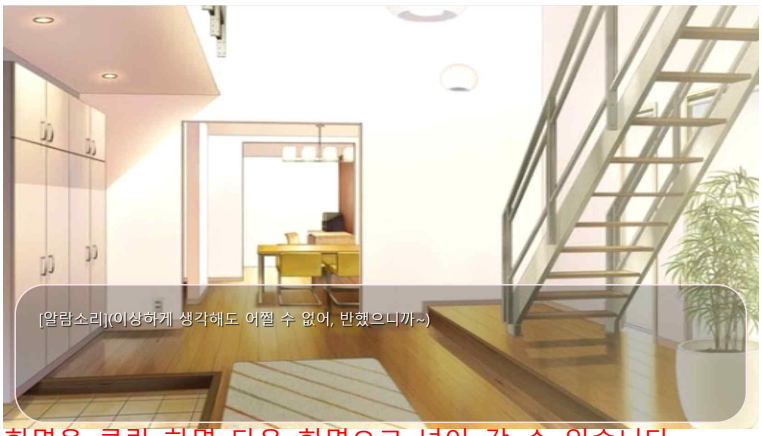
2. 작품 소개

가. 사용 설명서

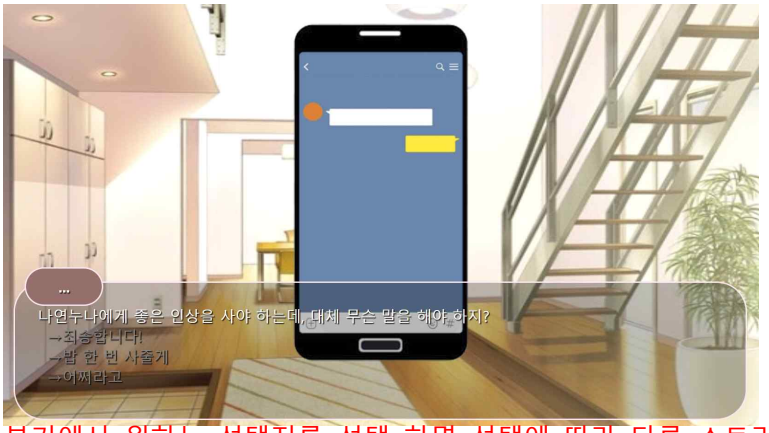
(각 메뉴 및 기능별 소스 등을 화면 캡쳐, 주요 기능 위주로 충분히 자세하게 설명할 것)

메인 페이지에서 URL 옆에 있는 버튼을 클릭하여 권한 탭의 자동 재생을 "오디오 및 비디오 허용"으로 설정 해 줍니다. 이 작업을 하지 않으면 소리가 나지 않습니다.

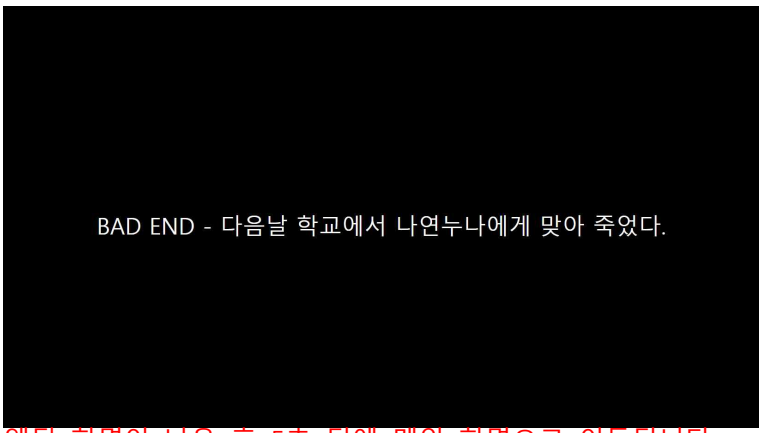
게임 시작 버튼을 클릭하면 게임 페이지로 넘어 가 집니다



화면을 클릭 하면 다음 화면으로 넘어 갈 수 있습니다.



분기에서 원하는 선택지를 선택 하면 선택에 따라 다른 스토리가 진행됩니다.



엔딩 화면이 나온 후 5초 뒤에 메인 화면으로 이동됩니다.

URL

나. 주요 소스코드

(주요기능 2-3가지 정도에 대해 자세히 설명)

Branch

대화 중 선택지가 나오는 부분입니다.

```
addBranch(branch) {
    this._branchManager.addBranch(branch);
}
```

Branch를 만들어 줍니다.

```
nextPage() {
    if (this._currentBranch.pages.length <= this._currentPageIndex)
        return;

    if (this._isBranching === true)
        return;

    if (this._currentPageIndex != 0)

    if (this._delaytimer !== null) {
        if (this._currentPageIndex != 0) {
            if (this._currentBranch.pages[this._currentPageIndex - 1].baseEvents.find(b => {
                return b.eventType === EventType.TextBar && b.textBarEventType === TextbarEventType.Branch
            }) != undefined)
                return;

            if (this._currentBranch.pages[this._currentPageIndex - 1].baseEvents.find(b => {
                return b.eventType === EventType.Canvas && b.canvasEventType === CanvasEventType.ChangeBackGround
            }) != undefined)
                return;

            if (this._currentBranch.pages[this._currentPageIndex - 1].baseEvents.find(b => {
                return b.eventType === EventType.Canvas && b.canvasEventType === CanvasEventType.RemoveObject
            }) != undefined)
                return;

            if (this._currentBranch.pages[this._currentPageIndex - 1].baseEvents.find(b => {
                return b.eventType === EventType.Sound
            }) != undefined)
                return;
        }

        clearTimeout(this._delaytimer);
        this._delaytimer = null;
    }

    //currentPage: Page
    let currentPage = this._currentBranch.pages[this._currentPageIndex];
    //currentEvents: BaseEvent[]
    let currentEvents = currentPage.baseEvents;

    this.eventProcess(currentEvents);

    if (this._currentBranch.pages.length <= this._currentPageIndex + 1) {
        let jumpBranch = this._branchManager.getBranch(this._currentBranch.end);
        if (jumpBranch !== null) {
            this._currentBranch = jumpBranch;
            this._currentPageIndex = -1;
        }
    }
    this._currentPageIndex += 1;
}
```

다음 페이지로 넘어 갈 때 Branch 관리를 해 줍니다.

```
branchProcess(branchPair, o) {
    let jumpBranch = o._branchManager.getBranch(branchPair.branch);
    if (jumpBranch == null) {
        o._canvasController.endScreen("END (Error)");
        o._isBranching = false;
        return;
    } else {
        o._currentBranch = jumpBranch;
        o._currentPageIndex = 0;
    }
    o._isBranching = false;
}
```

Branch 콜백 처리를 해 줍니다.

```
class BranchPair {
    //name: String, branch: Branch
    constructor(name, branch) {
        this._name = name;
        this._branch = branch;
    }

    //return: String
    get name() { return this._name; }

    //return: Branch
    get branch() { return this._branch; }
}
```

Branch의 이벤트 정보를 저장 해 줍니다.

```
class BranchManager {
    constructor() {
        this._branches = [];
    }

    //branch: Branch
    addBranch(branch) {
        this._branches.push(branch);
    }

    //branchName: String, return: Branch
    getBranch(branchName) {
        let find = this._branches.find(branch => branch.branchName === branchName);
        if (find === undefined)
            return null;
        else
            return find;
    }
}
```

Branch의 정보를 가져오거나 추가 해 줍니다.

```
class Branch {
    //branchName: String, end: String
    constructor(branchName, end) {
        this._branchName = branchName;
        this._end = end;
        //pages: Page[]
        this._pages = [];
    }

    //page: Page, return: Branch
    addPage(page) {
        this._pages.push(page);
        return this;
    }

    //name: String, text: String, return: Branch
    addTextPage(name, text) {
        this.addPage(new Page().addEvent(TextBarEvent.text(name, text)));
        return this;
    }

    //src: String, return: Branch
    addBackbroundPage(src){
        this.addPage(new Page().addEvent(CanvasEvent.changeBackGround(src)));
        return this;
    }

    //baseEvents: BaseEvent[]
    addEventsAsPage(baseEvents) {
        let page = new Page();
        baseEvents.forEach(element => {
            page.addEvent(element);
        });
        this.addPage(page);
        return this;
    }

    //text: String
    addEndingAsPage(text) {
        this.addPage(new Page().addEvent(CanvasEvent.showEnding(text)));
        return this;
    }

    //return: Page
    removePage() {
        return this._pages.pop();
    }

    //return: String
    get branchName() { return this._branchName; }
    //return: String
    get end() { return this._end; }
    //return: Page[]
    get pages() { return this._pages; }
}
```

Branch에서의 페이지, 사진, 텍스트 등을 추가 해 줍니다.

```
//options: BranchPair[], o: any, return: String
showBranch(options, o) {
    let button = [];
    let label = [];
    let text = [];
    for(let i = 0; i < options.length; i++){
        label[i] = document.createElement('label');
        button[i] = document.createElement('button');
        text[i] = document.createElement('span');
        this._chatBox.appendChild(label[i]);
        label[i].appendChild(button[i]);
        label[i].appendChild(text[i]);
        label[i].appendChild(document.createElement('br'));
        text[i].innerHTML = "→" + options[i].name;

        eventlistener(i).then((resolvedData) => {
            this._callback(resolvedData, o);
        });
    }
    let tmpThis = this;
    function eventlistener(i) {
        return new Promise(function(resolve, reject) {
            button[i].addEventListener('click', function(event) {
                resolve(options[i]);
                tmpThis.clearTextBar();
            }, false);
        })
    }
}
```

Branch를 화면에 띄우고, 선택지를 클릭 했을 때 정보를 전송 해 줍니다.

Canvas

화면에 이미지를 띄우는 부분입니다.

```
setCanvasElement(canvasImg, leftImg, centerImg, rightImg) {
    this._canvasController.init(canvasImg, leftImg, centerImg, rightImg);
}
```

CanvasController를 만들어 줍니다.

```
//canvasEvent: CanvasEvent
canvasProcess(canvasEvent) {
    switch (canvasEvent.canvasEventType) {
        case CanvasEventType.AddImage:
            this._canvasController.imageShow(
                canvasEvent._eventData.name,
                canvasEvent._eventData.src,
                canvasEvent._eventData.positon,
                canvasEvent._eventData.transition
            );
            break;

        case CanvasEventType.ChangeBackGround:
            this._canvasController.setBackground(
                canvasEvent._eventData.src
            );
            break;

        case CanvasEventType.DrawText:
            this._canvasController.drawtext(
                canvasEvent._eventData.text
            );
            break;

        case CanvasEventType.ShowEnding:
            this._canvasController.endScreen(
                canvasEvent._eventData.text
            );
            break;

        case CanvasEventType.RemoveObject:
            this._canvasController.imageRemove(
                canvasEvent._eventData.name,
                canvasEvent._eventData.transition
            );
            break;

        default:
            break;
    }
}
```

캔버스 이벤트 타입에 따라 CanvasController의 함수를 실행 해 줍니다.

```
const CanvasEventType = {
        AddImage: 0,
    ChangeBackGround: 1,
    DrawText: 2,
    ShowEnding: 3,
    RemoveObject: 4
}
```

캔버스 이벤트 타입 객체입니다.

```
class CanvasEvent extends BaseEvent {
    //canvasEventType: CanvasEventType
    constructor(canvasEventType, eventData) {
        super(EventType.Canvas);
        this._canvasEventType = canvasEventType;
        this._eventData = eventData;
    }

    //return: CanvasEvent
    static addImage(name, src, position, transition) {
        return new CanvasEvent(
            CanvasEventType.AddImage,
            new ImagePair(name, src, position, transition)
        );
    }

    //return: CanvasEvent
    static changeBackGround(src) {
        return new CanvasEvent(
            CanvasEventType.ChangeBackGround,
            new BackGroundPair(src)
        );
    }

    //return: CanvasEvent
    static drawText(text) {
        return new CanvasEvent(
            CanvasEventType.DrawText,
            new DrawTextPair(text)
        );
    }

    //return: CanvasEvent
    static showEnding(text) {
        return new CanvasEvent(
            CanvasEventType.ShowEnding,
            new DrawTextPair(text)
        );
    }

    //return: CanvasEvent
    static removeObject(name, transition) {
        return new CanvasEvent(
            CanvasEventType.RemoveObject,
            new RemoveObjectPair(name, transition)
        );
    }
```

캔버스 이벤트를 생성 해 줍니다.

```
class ImagePair {
    //name: String, src: String, position: modelPosition transition: String
    constructor(name, src, positon, transition) {
        this._name = name;
        this._src = src;
        this._positon = positon;
        this._transition = transition;
    }

    get name() { return this._name; }
    get src() { return this._src; }
    get positon() { return this._positon; }
    get transition() { return this._transition; }
}

class BackGroundPair {
    //src: String
    constructor(src) {
        this._src = src;
    }

    get src() { return this._src; }
}

class DrawTextPair {
    //text: String
    constructor(text) {
        this._text = text;
    }

    get text() { return this._text; }
}

class RemoveObjectPair {
    //name: String
    constructor(name, transition) {
        this._name = name;
        this._transition = transition;
    }
    get name() { return this._name; }
    get transition() { return this._transition; }
}
```

캔버스에 있는 오브젝트들의 정보를 제공합니다.

```
class CanvasController {
    constructor() {
        this._canvasImg = null;
        this._leftImg = null;
        this._centerImg = null;
        this._rightImg = null;
    }

    //canvasImg: HTMLElement, leftImg: HTMLElement, centerImg: HTMLElement, rightImg: HTMLElement
    init(canvasImg, leftImg, centerImg, rightImg) {
        this._canvasImg = canvasImg;
        this._canvasImg.style.opacity = "0";
        this._canvasImg.style.transitionDuration = "1s";
        this._leftImg = leftImg;
        this._centerImg = centerImg;
        this._rightImg = rightImg;
        let img = document.getElementById('canvasDiv').getElementsByClassName('img');
        for(let i = 0; i < 3; i++){
            img[i].style.opacity = 0;
        }
    }
    //name: String, src: String, positon: modelPosition transition: imageShowType
    imageShow(name, src, position, transition) { //
        let img;
        if(position === 0) {
            img = this._leftImg;
        }
        else if(position === 1) {
            img = this._centerImg;
        }
        else if(position === 2) {
            img = this._rightImg;
        }
        img.src = src;
        img.className = name;
        img.style.width = "40vw";
        if(transition === 1){

            img.style.transition = '2s';
            img.style.opacity = '1';
            //img.classList.add("show");
        }
        else{
            img.style.transition = 'all 0s';
            img.style.opacity = '1';
            //img.classList.add("showImmediately");
        }

    }
```

```
    //name: String, transition: imageHideType
    imageRemove(name, transition) {
        //name으로 이미지를 지울수 있음
        const img = document.getElementById('canvasDiv').getElementsByClassName(name)[0];
        if(img === undefined) return;
        img.className = "";
        if(transition === 1) {
            img.style.transition = '2s';
            img.style.opacity = '0';
            setTimeout(() => {
                img.src = "";
            }, 2000);
        }
        else{
            img.style.transition = '0s';
            img.style.opacity = '0';
            setTimeout(() => {
                img.src = "";
            }, 1000);
        }
    }

    //src: String
    setBackground(src) {
        this._canvasImg.style.opacity = "0";
        setTimeout(() => {
            this._canvasImg.src = src;
        }, 1000);
        setTimeout(() => {
            this._canvasImg.style.opacity = "1";
        }, 2000);
    }

    //text: String
    drawtext(text){ //null 넣으면 지워짐
        if(text === null){
            this._chatBox.innerHTML = "";
            return;
        }
        this._chatBox.innerHTML = text;
    }

    //text: String
    endScreen(text) {
        const canvasDiv = document.getElementById('canvasDiv');
        while(canvasDiv.firstChild){
            canvasDiv.removeChild(canvasDiv.firstChild);
        }

        canvasDiv.style.backgroundColor = "black";
        canvasDiv.classList.add("endScreen");
        canvasDiv.innerHTML = text;
        canvasDiv.style.color = "white";
        canvasDiv.style.paddingTop = "27vw";

        setTimeout(() => {
            location.href = "main.html";
        }, 5000);
    }
}
```

화면 위에 이미지를 생성하고, 지우고, 백그라운드 이미지를 변경하고, 엔딩 스크린을 생성합니다.

Text

대화 할 때 이름과 대화 내용을 출력합니다.

```
setTextBarElement(chatBox, nameBox) {
    this._textBarController.init(chatBox, nameBox);
}
```

textBarController를 만들어 줍니다.

```
//textBarEvent: TextBarEvent
textBarProcess(textBarEvent) {
    switch (textBarEvent.textBarEventType) {
        case TextbarEventType.Text:
            this._textBarController.setText(textBarEvent.eventData.name, textBarEvent.eventData.text);
            break;

        case TextbarEventType.Branch:
            this._isBranching = true;
            let selectedData = [];
            textBarEvent.eventData.forEach((item) => selectedData.push(item.name));
            this._textBarController.showBranch(textBarEvent.eventData, this);
            break;

        default:
            break;
    }
}
```

텍스트바 이벤트 타입에 따라 textBarController의 함수를 실행합니다.

```
//TextbarEventType: Number
const TextbarEventType = {
        Text: 0,
        Branch: 1
}
```

텍스트 바 이벤트 타입 객체입니다.

```
class TextBarEvent extends BaseEvent {
    //textBarEventType: TextBarEventType, eventData: any
    constructor(textBarEventType, eventData) {
        super(EventType.TextBar);
        //textBarEventType: TextBarEventType
        this._textBarEventType = textBarEventType;
        //eventData: any
        this._eventData = eventData;
    }

    //name: String, text: String, return: TextBarEvent
    static text(name, text) {
        return new TextBarEvent(TextbarEventType.Text, new TextPair(name, text));
    }

    //branchPairs: BranchPair[], return: TextBarEvent
    static branch(branchPairs) {
        return new TextBarEvent(TextbarEventType.Branch, branchPairs);
    }

    //return: TextbarEventType
    get textBarEventType() { return this._textBarEventType; }

    //return: any
    get eventData() { return this._eventData; }
}
```

텍스트 바 이벤트를 생성합니다.

```
class TextPair {
    //name: String, text: String
    constructor(name, text) {
        this._name = name;
        this._text = text;
    }

    //return: String
    get name() { return this._name; }

    //return: String
    get text() { return this._text; }
}
```

텍스트에 대한 정보를 제공합니다.

```
class TextBarController {
    constructor(callback) {
        this._chatBox = null;
        this._nameBox = null;
        this._callback = callback;
    }

    //chatBox: HTMLElement, nameBox: HTMLElement
    init(chatBox, nameBox) {
        this._chatBox = chatBox;
        this._nameBox = nameBox;
    }

    //name: String, text: String
    setText(name, text) {
        //const chatBox = document.getElementById("chatBox");
        if(name == null) {
            this._nameBox.style.visibility = "hidden";
            this._nameBox.innerHTML = "";
        } else{
            this._nameBox.style.visibility = "visible";
            this._nameBox.innerHTML = name;
        }
        this._chatBox.innerHTML = text + "<br>";
    }

    //options: BranchPair[], o: any, return: String
    showBranch(options, o) {
        let button = [];
        let label = [];
        let text = [];
        for(let i = 0; i < options.length; i++){
            label[i] = document.createElement('label');
            button[i] = document.createElement('button');
            text[i] = document.createElement('span');
            this._chatBox.appendChild(label[i]);
            label[i].appendChild(button[i]);
            label[i].appendChild(text[i]);
            label[i].appendChild(document.createElement('br'));
            text[i].innerHTML = "→" + options[i].name;

            eventlistener(i).then((resolvedData) => {
                this._callback(resolvedData, o);
            });
        }
        let tmpThis = this;
        function eventlistener(i) {
            return new Promise(function(resolve, reject) {
                button[i].addEventListener('click', function(event) {
                    resolve(options[i]);
                    tmpThis.clearTextBar();
                }, false);
            })
        }
    }

    clearTextBar() {
        while(this._chatBox.hasChildNodes()){
            this._chatBox.removeChild(this._chatBox.firstChild);
        }
        this._nameBox.style.visibility = "hidden";
        this._nameBox.innerHTML = "";
    }
}
```

텍스트 바에서 텍스트, 이름을 출력하고, 텍스트 바의 내용을 지우는 기능을 제공합니다.

다. 향후 작품 개선/발전 계획

응용 프로그램으로 구현되는 다른 비주얼 노벨들의 다른 기능을 우리의 웹 비주얼 노벨 엔진에도 많이 추가하는 것이 목표입니다.

3. 팀별 개인 역할

팀원	역할
팀원 1 이름: 김동희	엔진 구현, HTML, CSS 작성
팀원 2 이름: 김일중	기획, 스토리 작성, 사진 편집, 오디오 편집, 엔진에 이벤트 레지스터 등록
팀원 3 이름: 남현종	기획, 엔진 로직 디자인, 엔진에 이벤트 레지스터 등록

3. 느낀점

팀원 1(김동희):

항상 프레임워크를 많이 사용 해 와서 바닐라 자바스크립트 만으로 웹 게임이 잘 만들어 질까 하고 걱정도 했었는데, 원하는 대로 개발이 되어서 신기했고, 팀원들 간 소통과 분업도 잘 되고 모두가 열심히 한 덕에 만족스러운 결과물이 나온 것 같다.

팀원 2(김일중):

처음에 웹으로 개발을 해야 한다고 했을 때 에는 아는 게 없어 난처했는데, 팀원들의 도움으로 웹에 대해 배워가며 개발을 해낼 수 있었다.

개발을 끝마친 게임을 봤을 때 정보통신 시간에 배운 js, html, css만으로도 근사한 텍스트 게임을 만들 수 있다는 게 놀라웠다

팀원 3(남현종):

JS 에서 클린코드를 작성하는 것이 상당히 힘들었다.

팀원들이 전부 열심히 참여해주었다 다들 아침부터 새벽까지 16시간동안 같이 개발 해주었고 아마 이상적인 조별 프로젝트가 된 게 아닐까 싶다.