REPORT

과제명: 과제 #1. HTML Canvas

광운대학교
KwangWoon University

과목명	휴먼컴퓨터인터페이스
담당교수	이강훈 교수님
학과	전자융합공학과
학년	4학년
학번	2016742039
이름	정정현
제출일	21.04.10.

■개요

-요구조건/제약조건 만족도 및 추가 구현사항

1)요구조건(회색- **html 파일과 Canvas 요소 요구조건**, 하늘색- **Canvas 그리기 요구조건**, 연두색-**JS 객체지향프로그래밍 관련 요구조건**, 주황색- **추가 구현사항**)

요구조건	Canvas1	Canvas2	Canvas3	합계	만족도
하나의 .html 파일	결과물을 하나의 html 파일로 생성함				O
Canvas 요소 요구조건	O				O
2D 컨텍스트 정의	O				O
직선 객체 이용	18 개	8 개	3 개	29 개	O
원 객체 이용	32 개	0 개	1 개	33 개	O
사각형 객체 이용	14 개	3 개	13 개	30 개	O
다각형 객체 이용	0 개	8 개	1 개	9 개	O
베지어 곡선 객체 이용	8 개	5 개	0 개	13 개	O
텍스트 객체 이용	22 개	23 개	9 개	54 개	O
비트맵 이미지 객체 이용	8 개	6 개	9 개	23 개	O
화면당 객체의 수	102 개	53 개	35 개	-	O
JavaScript 이벤트 핸들러	화면 3 개 모두 서로 화면전환 가능				O
JavaScript 하나 이상의 객체 정의	**Polygon** 객체와 **ChattingBox** 객체 정의됨				O
JavaScript 시각적 표현	**Polygon, ChattingBox** 두 객체 모두 시각적 표현 적용됨				O
JavaScript 객체 2 개 이상 생성	**Polygon** 객체 9 개와 **ChattingBox** 객체 2 개 생성됨				O
JavaScript 대화형 상호작용	**ChattingBox** 객체로 대화형 상호작용 적용됨				O

2)제약조건

제약조건	만족도
클라이언트 측 스크립트만 사용	O
외부 라이브러리 및 리소스 사용 X	O
구글 크롬 웹 브라우저 호환	O

★주어진 요구조건/제약조건/추가구현 사항 모두 만족함

3)앱에서 선택된 3 개 화면&모사한 3 개 화면

▲앱에서 선택된 화면 (1)

▲모사한 화면 (1)

▲앱에서 선택된 화면 (2)

▲모사한 화면 (2)

▲앱에서 선택된 화면 (3)

▲모사한 화면 (3)

■본문

-소개(선택한 앱: 카트라이더 러쉬)

HTML Canvas 로 자주 사용하는 스마트폰의 어플 화면을 모사하라는 과제를 받고 제일 먼저 생각난 어플은 평소에 자주 즐겨하는 **카트라이더 러쉬**라는 게임 어플이었습니다. **카트라이더 러쉬**는 다른 어플들에 비해 많은 Activity(화면)를 갖고 있고 한 Activity(화면)마다 모사가 가능해 보이는 다양한 요소들이 굉장히 많아 보이기 때문에 과제에서 주어진 요구조건과 제약조건을 충분히 만족시킬 수 있겠다고 판단해 **카트라이더 러쉬**를 모사할 어플로 선정하게 되었습니다.

어플은 앞서 말했듯이 다양한 요소들이 있어 어떤 요소들은 표현하기 수월한 반면 어떠한 요소들은 Canvas 로 표현하기 어려운 부분이 존재했습니다. 그래서 표현하기 수월한 요소들(배경화면, 메뉴, 게임시작 버튼 등...)은 최대한 Canvas 의 2D context 로 모사를 하고 어려운 요소들(자동차, 메뉴의 작은 아이콘, 등...)은 resource 폴더에 해당 이미지들을 저장한 후 비트맵 이미지를 이용해 모사를 하기로 결정했습니다.

-Canvas 그리기

Canvas 는 화면 전환을 위해 한 화면 마다 한 개의 함수를 할당해 함수 안에 해당하는 화면을 그리는 코드를 작성하였습니다. 어플 3 개의 화면을 모사해야 하므로 Canvas 를 그리는 함수는 **create_canvs_main(), create_canvas_storage(), create_canvas_gamestart()** 이렇게 3 가지가 있습니다.

1) create_canvas_main()

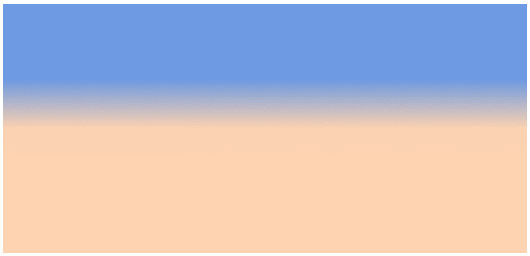
create_canvas_main()함수에서 모사할 어플의 화면은 아래와 같습니다.

a)배경

먼저 가장 뒤쪽에 있는 요소를 먼저 그려야 하기 때문에 배경을 그리기로 결정했습니다. 배경은 context.fillRect()를 활용해 그렸습니다. 그런데, 현재 원본 어플의 배경은 굉장히 많은 요소들이 있고 이것을 다 표현하기에는 어려움이 있을 것 같다고 생각해 하늘과 땅의 색상 코드를 얻어 그라데이션 형식으로 표현하고 나머지 복잡한 요소들을 생략하기로 결정했습니다.

```
var background_color = context.createLinearGradient(0,0,0,canvas.height);
            background_color.addColorStop(0.3,'#6D9AE2');/*배경 윗쪽 하늘*/
            background_color.addColorStop(0.5,'#FDD3B2');/*배경 아래쪽 땅*/
            context.fillStyle=background_color;
            context.fillRect(0,0,canvas.width,canvas.height);
            /*mainCanvas 배경 그라데이션으로 채우기*/
```

하늘과 땅의 색상 코드를 포토샵을 통해 얻고 addColorStop() 함수를 이용해 색상의 비율과 코드를 background_color 변수에 넣은 뒤 fillStyle 을 통해 Rect 의 색상의 지정하고 fillRect 의 범위를 canvas 를 가득 채우도록 하게 해 배경을 생성해 주어 아래 그림과 같은 배경을 만들어 주었습니다.



▲create_canvas_main()의 배경 예시

b)하단 메뉴바, 시작버튼, 채팅창

원본 어플의 배경을 보면 구름이나 풀과 같은 요소가 있기 때문에 이 요소들은 2D context 의 원을 이용해 표현했습니다. 원을 생성하는 코드는 추후에도 많이 사용될 것이라 생각되어 코드를 function 으로 따로 만들어 사용하였습니다.

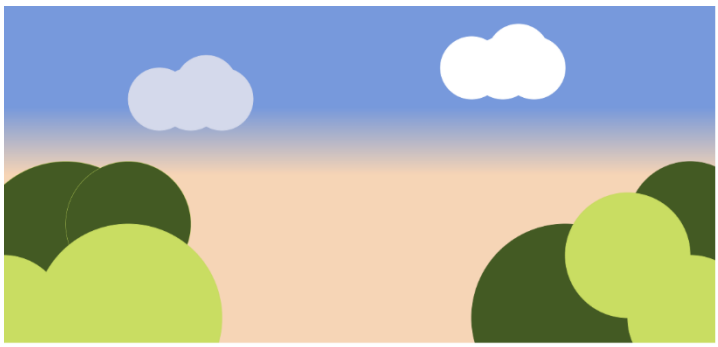
```
function create_circle(x,y,size,line_width){
            context.lineWidth = line_width;
            context.beginPath();
            context.arc(x,y,size,0,Math.PI*2);
            context.stroke();
            context.fill();
        };/*원 그리기 func*/
```

▲원을 생성하는 함수

```
context.fillStyle = 'white';
context.strokeStyle ='white';
create_circle(800,100,50);
create_circle(850,100,50);
create_circle(750,100,50);
create_circle(825,80,50);
context.fillStyle = '#D3D9ED';
context.strokeStyle ='#D3D9ED';
create_circle(300,150,50);
create_circle(350,150,50);
create_circle(250,150,50);
create_circle(325,130,50);/*구름 표현*/
```

▲구름을 canvas 에 그리는 코드

위에 있는 코드와 같이 context 의 fillStyle 과 strokeStyle 을 지정해 주고 원들의 중심 좌표를 다른 원들과 겹치게 만들어 구름처럼 보이도록 코드를 작성하였습니다. 구름 외에도 화면 하단 양쪽 풀을 표현한 코드도 거의 동일함으로 생략하겠습니다.



▲위의 코드와 같은 방식으로 구름, 풀이 예시

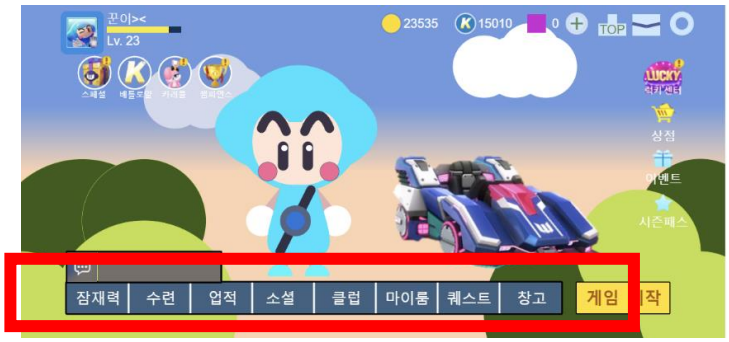
배경을 다음으로 어플 메인 화면의 하단에 있는 메뉴바는 2D context 의 사각형과 직선, 텍스트를 활용해 모사하였습니다. 메뉴바의 색상 같은 경우는 앞선 경우와 똑같이 포토샵을 이용해 색상 코드를 따와서 fillStyle 에 지정해 주어 사용하였습니다.

```
context.fillStyle = '#3E6282';
context.strokeStyle='black';
context.lineWidth = 5;
context.strokeRect(75,450,800,50);
context.fillRect(75,450,800,50);/*x,y,width,height*/
context.fillStyle = 'white';
context.font = '400 18pt Arial';
context.textAlign = 'start';
```

```
            var down_menu_index = ["잠재력","  수련","  업적"," 소셜","  클럽","
마이룸","퀘스트","  창고"],
                index_count=0,
                chat_img = new Image();
            for(var i =175;i<800;i+=100){
                context.lineWidth = 3;
                context.strokeStyle='white';
                context.beginPath();
                context.moveTo(i,450);
                context.lineTo(i,450+50);
                context.stroke();
                context.fillText(down_menu_index[index_count++],i-85,485);
            }
            context.fillText(down_menu_index[7],875-85,485);
              /*하단 메뉴 바&text*/
```

▲구름을 canvas 에 그리는 코드

메뉴바는 좌표와 크기를 지정해 주고 context 의 strokeRect()와 fillRect()를 통해 생성해 주었습니다. 그리고 메뉴바는 배열과 같이 각각의 메뉴버튼들이 직선으로 나누어져 있음으로 메뉴 버튼들의 텍스트는 down_menu_index 배열로 저장을 해주고 for 문을 이용해 직선을 일정 간격(var i)으로 반복해서 text 와 직선을 그리게 코드를 작성하였습니다.



▲빨간색 네모 안에 메뉴바가 직선과 텍스트와 함께 표현된 예시

채팅창의 말풍선 도형은 모사하기 복잡한 부분이 있다고 판단해 포토샵을 이용해 말풍선 이미지를 resource 폴더에 저장하여 말풍선을 모사하는데 활용했습니다.

```
            context.fillStyle = '#786F68';
            context.strokeStyle = 'black';
            context.fillRect(125,400,200,50);
            context.strokeRect(75,400,250,50);
            chat_img.src = 'resource/chatImg.png';
            create_Image(chat_img,76,401,48,48);
```

코드는 채팅이 입력될 부분과 말풍선 이미지를 나눠서 작성하였습니다. 채팅이 입력될 부분은 fillRect()와 strokeRect()로 사각형으로 생성해 주었고 말풍선 같은 경우는 Image()객체로 지정된 변수인 chat_img 의 src 를 저장해 놓은 말풍선 이미지로 지정하고 create_Image() 함수를 통해 생성해 주었습니다.

Cf. create_Image() 함수는 앞선 create_circle()과 같이 자주 사용될 것 같다고 판단되어 이미지를 생성하는 코드를 아래와 같이 함수로 만들어 주었습니다.

```
function create_Image(img,x,y,w,h){
            img.onload=function(e){
                context.drawImage(img,x,y,w,h);
            }
        };/*비트맵 이미지 생성*/
```

▲비트맵 이미지를 생성하는 함수 코드

함수의 내용으로는 img parameter 에 Image() 객체를 받고 이미지를 그릴 좌표는 x, y 에 너비와 높이를 w, h 에 받아 이미지를 생성해 주는 코드입니다.

▲빨간색 네모 안 채팅창의 예시

```
            context.fillStyle='#FFDF20';
            context.strokeStyle='black';
            context.strokeRect(900,450,150,50);
            context.fillRect(900,450,150,50);
            context.font = '700 20pt Arial';
            context.fillStyle='#AA5905';
            context.fillText("게임  시작",915,485);
            /*게임 시작버튼*/
```

▲게임 시작 버튼 생성하는 코드

게임 시작 버튼은 앞선 메뉴바와 비슷하게 strokeRect()와 fillRect()로 사각형을 그려주고 그 위에 fillText 를 통해 text 를 생성해 만들었습니다.

◀게임시작버튼의 예시

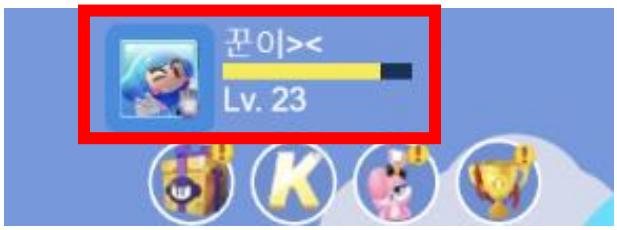
c)화면 상단 인터페이스

```
var user_profile_img=new Image();
user_profile_img.src='resource/userProfile.png';
context.lineJoin='round';
context.lineWidth='20';
context.strokeStyle='#2D8EE2';
context.strokeRect(75,20,50,50);
create_Image(user_profile_img,75,20,50,50);
context.font = '300 15pt Arial';
context.fillStyle = 'white';
context.fillText("꾼이><",140,30);
context.fillText("Lv. 23",140,65);
context.fillStyle='#14315A';
context.fillRect(140,35,120,10);
context.fillStyle='#F7EB35';
context.fillRect(140,35,100,10);
/*화면 좌측 상단 user 아이콘,경험치 바*/
```

▲화면 좌측 상단 인터페이스 생성 코드

화면 좌측 상단의 user 의 아이콘은 옅은 색의 사각형으로 둘러 쌓인 아이콘입니다. 이 아이콘을 모사하기위해 context 의 lineJion 을 round 로 지정해주고 strokeRect()를 통해 모서리가 둥근 옅은 색의 사각형을 생성하고 Image() 객체를 생성해 resource 폴더의 userProfile.png 를 지정한 후 create_Image() 함수를 통해 아이콘을 모사해 주었습니다.

아이콘 옆의 아이디, 레벨은 context 의 font 를 두께, 크기, 폰트 순으로 지정해주고 fillStyle()을 통해 색상을 정한 후 fillText()로 각각의 텍스트를 표현했습니다. 경험치 바는 다른 색상을 가진 두개의 사각형을 fillRect()로 생성하고 width 만 다르게 해 원본 어플의 경험치 바를 모사하였습니다.

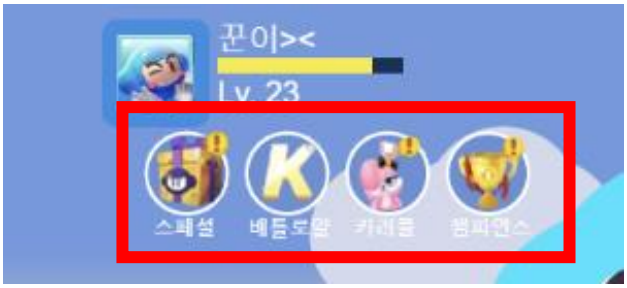


▲화면 좌측 상단 인터페이스 예시

```
            var upper_menu_index = [" 스페셜","배틀로얄"," 카러플","챔피언스"],
                index_count2=0,img_count=0,
                upper_menu_img1 = new Image(),upper_menu_img2 = new Image(),
                upper_menu_img3 = new Image(),upper_menu_img4 = new Image();
            upper_menu_img1.src = 'resource/upperMenuImg1.png';
            upper_menu_img2.src = 'resource/upperMenuImg2.png';
            upper_menu_img3.src = 'resource/upperMenuImg3.png';
            upper_menu_img4.src = 'resource/upperMenuImg4.png';

            for(var circle_i=120;circle_i<320;circle_i+=65){
                context.fillStyle='#6690D5';
                context.strokeStyle='white';
                create_circle(circle_i,110,25,5);
                context.fillStyle='white';
                context.strokeStyle='#6690D5';
                context.font='400 10pt Arial';
                context.fillText(upper_menu_index[index_count2++],circle_i-
25,150);
            }/*화면 상단 원 4개 메뉴*/
            create_Image(upper_menu_img1,90,80,60,60);
            create_Image(upper_menu_img2,155,80,60,60);
            create_Image(upper_menu_img3,220,80,60,60);
            create_Image(upper_menu_img4,285,80,60,60);
            /*화면 상단 원 4개 메뉴 이미지*/
```

▲화면 좌측 상단 메뉴 생성 코드

화면 좌측 상단 메뉴는 텍스트와 원, 비트맵 이미지로 생성을 했습니다. 원과 텍스트는 for 문을 이용해 x 축 값만 증가시켜 일정 간격을 갖으면서 반복적으로 4 개의 원과 텍스트를 생성하게 코드를 작성했습니다. 원안에 들어갈 이미지들을 각각의 Image()객체를 만들어 src 를 지정해주고 원 안의 좌표를 지정해 create_Image() 함수로 생성해주었습니다.

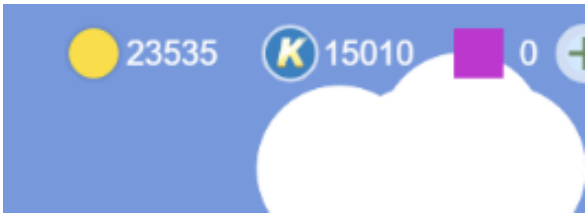


▲좌측 상단 메뉴의 예시

```
context.fillStyle = '#FFDC05';
create_circle(600,30,15,5);
context.fillStyle = 'white';
context.font = '400 15pt Arial';
context.fillText("23535",620,37);
```

▲화면 상단 코인 생성 코드

화면 상단의 코인들은 위 코드와 같이 fillStyle 로 색상을 지정하고 create_circle() 함수를 통해 원을 생성한 뒤 옆에다 fillText()로 text 를 생성해주었습니다. 다른 코인들은 위 코드가 반복되는 내용이므로 생략하겠습니다.

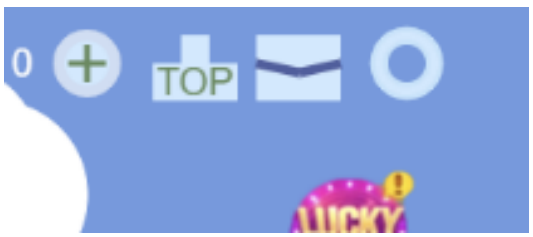


▲화면 상단 코인들의 예시

```
context.fillStyle='#CEE9FF';
create_circle(900,30,12,12,5);
context.strokeStyle = '#638A5D';
context.lineWidth=3;
context.beginPath();
context.moveTo(890,30);
context.lineTo(910,30);
context.stroke();
context.beginPath();
context.moveTo(900,20);
context.lineTo(900,40);
context.stroke();
```

▲화면 우측 상단 아이콘 생성 코드

화면 우측 상단의 +, 편지, 설정 아이콘은 위 코드와 같이 원이나 사각형을 생성하고 직선을 생성하는 코드를 이용해 아이콘들을 생성해 주었습니다. 다른 아이콘들은 위 코드와 비슷한 내용이므로 생략하겠습니다.



◀화면 우측 상단 아이콘의 예시

d)화면 중앙 캐릭터, 카트

```
            var cart_img = new Image();
            cart_img.src = 'resource/cart.png';
            create_Image(cart_img,600,230,350,200);/*화면 중앙 카트*/
```

▲화면 중앙 카트 생성 코드

화면 중앙의 카트는 표현하기 복잡한 부분이 많아 create_Image() 함수를 활용해 비트맵 이미지로 표현하였습니다.

▲화면 중앙 카트 예시

화면 중앙에 있는 캐릭터는 원과 베지어 곡선, 직선, 사각형을 통해 표현을 했습니다. 베지어 곡선을 생성하는 코드는 자주 사용될 것이라 생각되고 곡선을 생성하는 코드가 길어 함수를 생성해 사용을 하였습니다.

```
function create_bezierCurve(stk_style,end_point,control_point,choosefill){
            context.strokeStyle = stk_style;
            context.fillStyle = stk_style;
            context.beginPath();
            context.moveTo(end_point[0].x,end_point[0].y);
            context.bezierCurveTo(control_point[0].x,control_point[0].y
                                  ,control_point[1].x,control_point[1].y
                                  ,end_point[1].x,end_point[1].y);
            context.stroke();
            if(choosefill==1){
                context.fill();
            }
        };/*베지어 곡선 그리기 func*/
```

▲베지어 곡선 생성 함수

함수의 parameter 로는 곡선의 색상과 양 끝점을 표현하는 배열, 제어점을 표현하는 배열 그리고 곡선 내부를 채울지 말지를 정하는 변수로 받았습니다.

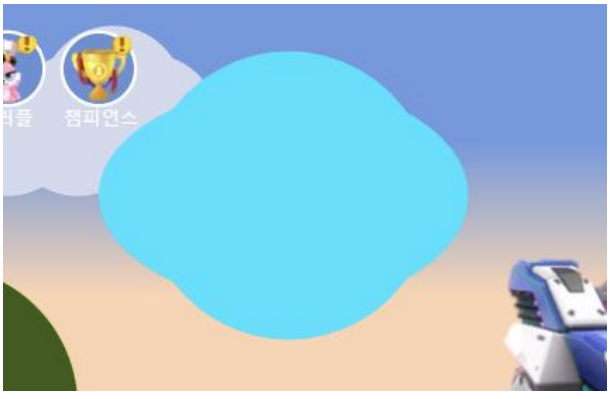
그 뒤 입력 받은 parameter 에 따라 베지어 곡선을 생성하는 context 의 bezierCurveTo() 함수를 이용해 곡선을 그리는 함수입니다.

캐릭터를 생성하는데 많은 2D context 가 사용되어 캐릭터의 신체를 나눠서 설명하겠습니다.

```
            context.fillStyle = '#21E1FF';
             context.strokeStyle ='#21E1FF';
             create_circle(450,200,100);/*다오 얼굴*/
             var dao_head_left_end_point = [{x:400,y:130},{x:400,y:270},],
/*시작점, 끝점*/
                 dao_head_left_control_point = [{x:290,y:160},{x:290,y:240},],
/*중간 control Point*/
                 dao_head_right_end_point = [{x:500,y:130},{x:500,y:270},],
                 dao_head_right_control_point = [{x:600,y:160},{x:600,y:240},],
                 dao_body_left_end_point = [{x:430,y:300},{x:450,y:400},],
                 dao_body_left_control_point = [{x:400,y:400},{x:400,y:400},],
                 dao_body_right_end_point = [{x:470,y:300},{x:450,y:400},],
                 dao_body_right_control_point = [{x:500,y:400},{x:500,y:400},],
                 dao_face_left_end_point = [{x:430,y:200},{x:460,y:280},],
                 dao_face_left_control_point = [{x:380,y:130},{x:350,y:300},],
                 dao_face_right_end_point = [{x:460,y:200},{x:440,y:280},],
                 dao_face_right_control_point = [{x:510,y:130},{x:550,y:300},],
                 dao_eye_left_end_point = [{x:430,y:205},{x:390,y:200},],
                 dao_eye_left_control_point = [{x:430,y:205},{x:410,y:150},],
                 dao_eye_right_end_point = [{x:460,y:205},{x:500,y:200},],
                 dao_eye_right_control_point = [{x:460,y:205},{x:480,y:150},];
             create_bezierCurve('#21E1FF',dao_head_left_end_point,dao_head_left
_control_point,1);
             create_bezierCurve('#21E1FF',dao_head_right_end_point,dao_head_rig
ht_control_point,1);/*다오 머리*/
```

▲캐릭터 머리 생성 코드

캐릭터의 머리는 create_circle()을 통해 머리를 생성하고 캐릭터는 곡선으로 표현해야 할 부분이 많아 각 곡선들에 해당되는 적절한 끝점과 제어점을 미리 다 지정해 주었습니다. 캐릭터의 머리를 보면 양쪽에 곡선으로 생성된 부분이 있음으로 이를 표현하기 위해 create_bezierCurve() 함수를 사용해 좌우측 곡선으로 된 부분을 생성해 주었습니다.

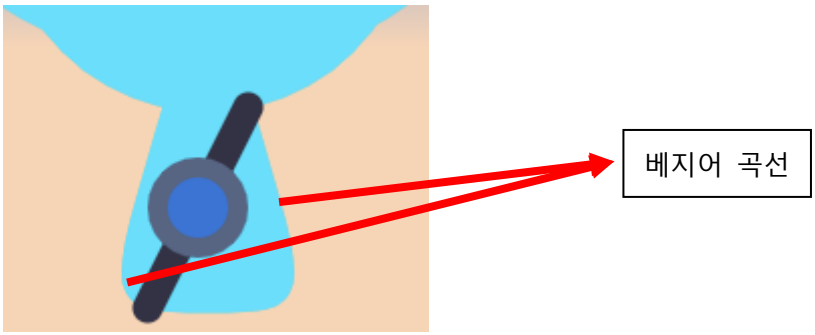


◀캐릭터의 머리 예시

```
create_bezierCurve('#21E1FF',dao_body_left_end_point,dao_body_left_control_poi
nt,1);
            create_bezierCurve('#21E1FF',dao_body_right_end_point,dao_body_rig
ht_control_point,1);
            context.fillRect(430,300,40,100);/*다오 몸통*/
            context.strokeStyle = '#333247';
            context.lineWidth = 15;
            context.lineCap = 'round';
            context.beginPath();
            context.moveTo(470,300);
            context.lineTo(420,400);
            context.stroke();
            context.strokeStyle = '#536585';
            context.fillStyle = '#2276DB';
            create_circle(445,350,15,20);/*다오 밸트*/
```

▲캐릭터 몸통 생성 코드

캐릭터의 몸통도 곡선 형태로 이루어져 있어 create_bezierCurve() 함수를 사용해 몸통 좌 우측을 생성해 주고 가운데 비어있는 부분을 fillRect()를 통해 사각형을 생성해 채워주었습니다. 그리고 캐릭터가 착용하고 있는 벨트는 context 의 lineWidth 를 15 로 지정하고 lineCap 은 round 로 지정해 둥글고 굵은 직석을 표현하게 해 만든 뒤 create_circle() 함수로 원을 생성해 벨트 가운데 보석을 표현해 주었습니다.



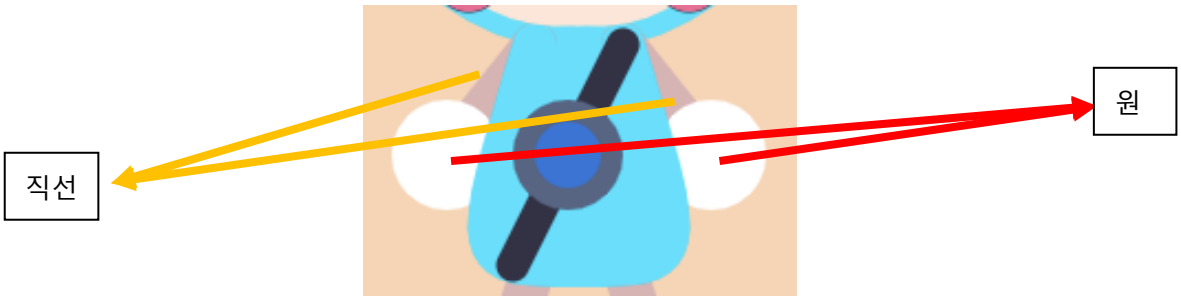
▲캐릭터 몸통 예시

```
            context.lineWidth = 20;
            context.strokeStyle ='#DEB2B3';
            context.lineCap = 'round';
            context.beginPath();
            context.moveTo(430,300);
            context.lineTo(390,350);
            context.stroke();
            context.beginPath();
            context.moveTo(470,300);
            context.lineTo(510,350);
```

```
            context.stroke();
            context.fillStyle = 'white';
            context.strokeStyle ='white';
            create_circle(390,350,15,20); create_circle(510,350,15,20);
              /*다오 팔*/
```

▲캐릭터 팔 생성 코드

캐릭터의 팔은 앞서 캐릭터의 몸통에 있는 벨트를 만들 때와 같이 context 의 lineWidth 를 두껍게 하고 lineCap 을 round 로 설정해 끝이 둥글고 두꺼운 직선을 그리게 한 뒤 팔의 끝 좌표에 create_circle()로 원을 생성해 팔을 생성해 주었습니다.



▲캐릭터 팔 예시

```
context.lineWidth = 20;
            context.strokeStyle ='#DEB2B3';
            context.lineCap = 'round';
            context.beginPath();
            context.moveTo(430,400);
            context.lineTo(420,430);
            context.stroke();
            context.beginPath();
            context.moveTo(470,400);
            context.lineTo(480,430);
            context.stroke();/*다오 다리*/

            context.fillStyle = '#31F6FF';
            context.strokeStyle ='white';
            context.beginPath();
            context.arc(420,440,20,Math.PI,false);
            context.fill();
            context.beginPath();
            context.arc(480,440,20,Math.PI,false);
            context.fill();/*다오 발*/
```

▲캐릭터 다리&발 생성 코드

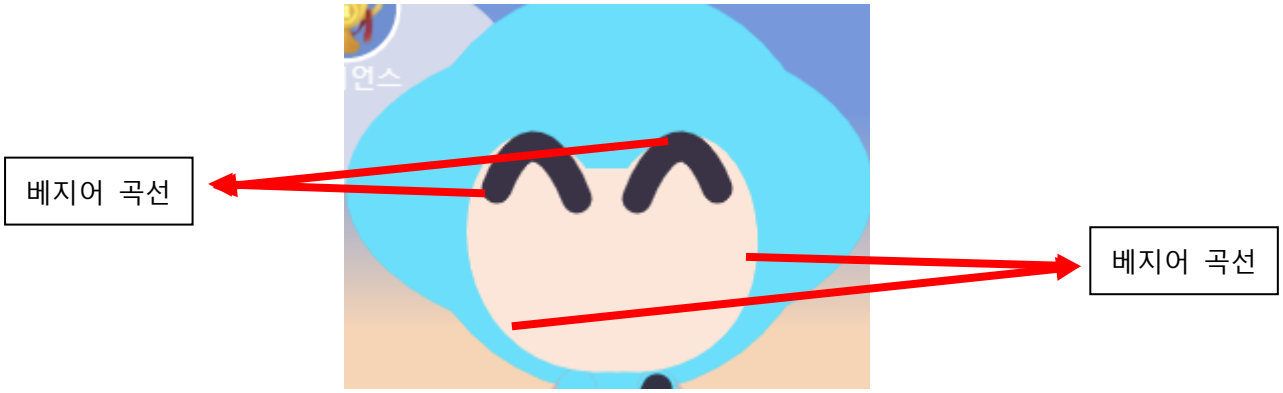
캐릭터의 다리는 앞선 팔과 같은 방식으로 생성하였습니다. 캐릭터의 발의 경우 열린 원호를 생성하는 방법인 context 의 arc()를 이용해 원호의 시작점을 Math.PI 로 설정하고 원호를 그리는 방향을 false 를 통해 반시계 방향으로 돌려 아래가 평평한 반원을 생성해 발을 표현하였습니다.

▲캐릭터 다리&발의 예시

```
create_bezierCurve('#FFE5D7',dao_face_left_end_point,dao_face_left_control_poi
nt,1);
            create_bezierCurve('#FFE5D7',dao_face_right_end_point,dao_face_rig
ht_control_point,1);
            context.fillStyle ='#FFE5D7';
            context.fillRect(430,190,30,70);/*다오 얼굴*/
            context.lineWidth = 15;
            create_bezierCurve('#3A3346',dao_eye_left_end_point,dao_eye_left_c
ontrol_point);
            create_bezierCurve('#3A3346',dao_eye_right_end_point,dao_eye_right
_control_point);/*다오 눈썹*/
```

▲캐릭터 얼굴 테두리&눈썹 생성 코드

캐릭터의 원본을 살펴보면 얼굴 테두리가 하트 모양과 비슷한 모양로 되어있고 눈썹도 곡선을 띄고 있어 얼굴 테두리와 눈썹을 create_bezierCurve() 함수를 사용해 기존에 생성해 놓은 캐릭터의 머리위에 얼굴 테두리와 눈썹을 생성하였습니다. 그리고 두개의 베지어 곡선을 그리고 나니 가운데 부분이 비게 되어 fillRect()로 사각형을 생성해 비어 있는 내부를 채워주었습니다.



▲캐릭터 얼굴 테두리&눈썹의 예시

```
            context.lineWidth = 17;
            context.beginPath();
            context.moveTo(425,230);
            context.lineTo(425,260);
            context.stroke();
            context.beginPath();
            context.moveTo(465,230);
            context.lineTo(465,260);
            context.stroke();
            context.fillStyle = 'white';
            create_circle(427,232,5,1);
            create_circle(467,232,5,1);
            context.fillStyle='#F56796';
            create_circle(400,270,15,1);
            create_circle(500,270,15,1);/*다오 얼굴 내부*/
```

▲캐릭터 얼굴 내부 생성 코드

캐릭터의 얼굴 내부는 두개의 눈과 눈동자, 홍조가 있습니다. 눈은 다리와 팔을 만들었을 때와 똑같이 직선을 사용해 생성해 주었고 눈동자와 홍조는 create_circle() 함수를 활용해 원을 생성하여 표현했습니다.

▲캐릭터 얼굴 내부 예시

e)화면 우측 메뉴바

```
var right_menu_img = new Image(),
            right_menu_txt = ["상점" ,"이벤트","시즌패스"],
            right_menu_count = 0;
            right_menu_img.src = 'resource/rightMenu.png';
            create_Image(right_menu_img,1000,80,80,300);
            context.fillStyle='white';
            context.textAlign = 'center';
            context.font = '400 15pt Arial';
            for(var right_menu_index = 220;right_menu_index<420;right_menu_ind
ex+=70){
                context.fillText(right_menu_txt[right_menu_count++],1040,right
_menu_index);
            }
```

▲화면 우측 메뉴바 생성 코드

화면 우측 메뉴바는 앞서 메뉴바를 생성했던 것과 비슷한 방식으로 먼저 Image() 객체의 src 에 메뉴 이미지를 지정해 주고 이미지를 생성한 뒤 for 문을 이용해 y 축을 일정하게 증가시켜 right_menu_txt 배열에 있는 text 를 일정 간격으로 반복해서 생성시켰습니다. 또한 text 는 가운데 위치하는 것이 원본 어플과 유사하므로 context 의 textAlign 을 center 로 지정해 text 가 가운데로 정렬되어 생성될 수 있게 했습니다.

◀화면 우측 메뉴바 예시

2) create_canvas_storage()

create_canvas_storage()함수에서 모사할 어플의 화면은 아래와 같습니다.

a)배경

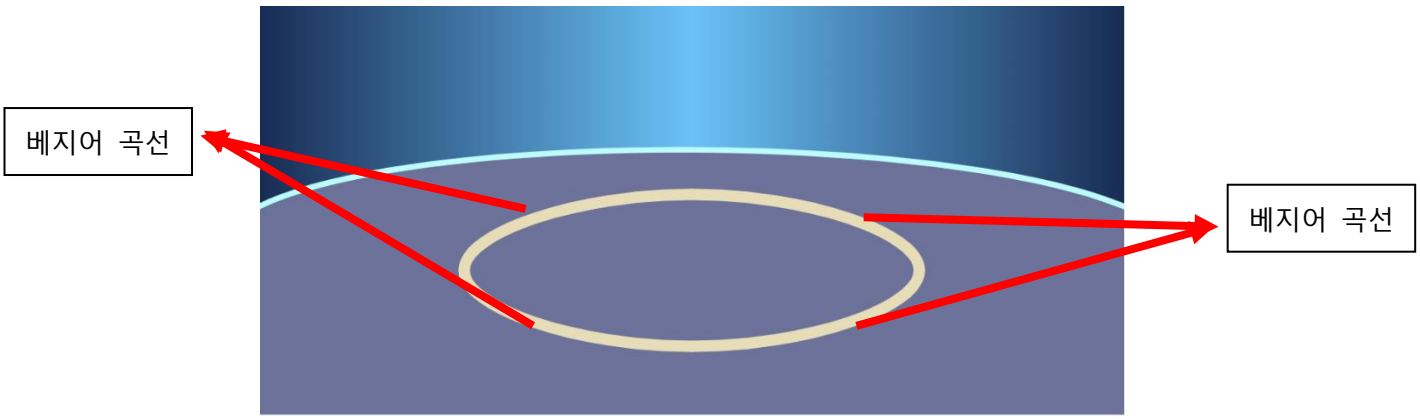
```
var storage_background_color = context.createLinearGradient(0,0,canvas.width,0
);
            storage_background_color.addColorStop(0,'#122D57');
            storage_background_color.addColorStop(0.5,'#42C5FE');
            storage_background_color.addColorStop(1,'#122D57');
            context.fillStyle=storage_background_color;
            context.fillRect(0,0,canvas.width,canvas.height);
              /*배경 그라데이션 효과*/

            var storage_background_center_end_point = [{x:0,y:canvas.height/2}
,{x:canvas.width,y:canvas.height/2},],
                storage_background_center_control_point=[{x:200,y:canvas.heigh
t/2-100},{x:900,y:canvas.height/2-100},],
                center_ellipse_1_end_point =[{x:canvas.width/2,y:250},{x:canva
s.width/2+300,y:350},],
                center_ellipse_1_control_point =[{x:canvas.width/2+150,y:250},
{x:canvas.width/2+300,y:300},],
                center_ellipse_2_end_point =[{x:canvas.width/2+300,y:350},{x:c
anvas.width/2,y:450},],
                center_ellipse_2_control_point =[{x:canvas.width/2+300,y:400},
{x:canvas.width/2+150,y:450},],
                center_ellipse_3_end_point =[{x:canvas.width/2,y:450},{x:canva
s.width/2-300,y:350},],
                center_ellipse_3_control_point =[{x:canvas.width/2-
150,y:450},{x:canvas.width/2-300,y:400},],
                center_ellipse_4_end_point =[{x:canvas.width/2-
300,y:350},{x:canvas.width/2,y:250},],
                center_ellipse_4_control_point =[{x:canvas.width/2-
300,y:300},{x:canvas.width/2-150,y:250},];
            context.lineWidth = 15;
            create_bezierCurve('#ACFFFF',storage_background_center_end_point,s
torage_background_center_control_point);
            context.fillStyle ='#6B729C';
            context.fill();
            context.fillRect(0,canvas.height/2,canvas.width,canvas.height/2);
            create_bezierCurve('#E9DEB4',center_ellipse_1_end_point,center_ell
ipse_1_control_point);
            create_bezierCurve('#E9DEB4',center_ellipse_2_end_point,center_ell
ipse_2_control_point);
            create_bezierCurve('#E9DEB4',center_ellipse_3_end_point,center_ell
ipse_3_control_point);
            create_bezierCurve('#E9DEB4',center_ellipse_4_end_point,center_ell
ipse_4_control_point);/*뒷 배경*/
```

▲화면 배경 생성 코드

create_canvas_storage()의 배경은 크게 원본 어플 가운데에 위치하는 곡선을 중심으로 위쪽과 아래쪽으로 나눠서 두개 부분으로 나눠서 봤습니다. 곡선 기준 위쪽 부분은 storage_background_color 변수를 생성해 addColorStop 으로 배경 가운데 색상과 양쪽 색상을 넣은 후 가운데 색상으로부터 양쪽으로 그라데이션이 퍼져 나가는 방식으로 표현하고 fillRect()를 통해 위쪽 배경을 생성하였습니다.

▲양쪽으로 퍼지는 그라데이션으로 표현한 배경 예시

곡선을 기준으로 위쪽 배경과 아래쪽 배경을 나눴음으로 곡선을 create_bezierCurve() 함수를 호출해 곡선을 생성해 주고 아래쪽 배경을 채우기 위해 fillStyle 을 원본 어플의 아래쪽과 같은 색상으로 지정하고 fillRect()를 통해 아래쪽 배경을 생성하였습니다. 그리고 원본 어플을 보면 카트 아래 타원형으로 된 원이 있는데 타원 같은 경우는 center_ellipse 라는 끝점과 제어점을 갖은 베지어 곡선 4 개의 모서리를 연쇄적으로 이어 붙이는 것을 이용해 생성했습니다.



▲화면 위아래 구분하는 곡선&카트가 올라갈 타원을 표현한 배경 예시

b)화면 좌측 메뉴바&화면 중앙 카트

```
var storage_cart_img = new Image(),
                storage_menu_count = 0, storage_menu = ["카트","캐릭터","플라잉
 펫","펫","엠블럼","아이템","컬렉션"];
            storage_cart_img.src = 'resource/storageCart.png';
            create_Image(storage_cart_img,300,160,520,270);/*화면 중앙 카트*/

            context.fillStyle = '#215A9F';
            context.fillRect(0,0,200,canvas.height);
            context.fillStyle = 'white';
            context.font = '400 15pt Arial';
            for(var storage_left_menu_i = 50;storage_left_menu_i<canvas.height
;storage_left_menu_i+=70){
                context.lineWidth = 2;
                context.strokeStyle='white';
                context.beginPath();
                context.moveTo(0,storage_left_menu_i);
                context.lineTo(200,storage_left_menu_i);
                context.stroke();
                context.fillText(storage_menu[storage_menu_count++],130,storag
e_left_menu_i+40);
            }/*창고 좌측 메뉴*/
```

▲화면 좌측 메뉴바&중앙 카트 생성 코드

화면 좌측 메뉴바는 앞서 create_canvas_main() 함수에서 하단 메뉴바를 생성했던 것과 같은 원리로 메뉴바로 사용할 사각형을 fillRect()함수를 통해 생성하고 for 문을 이용해 메뉴 index 를 나누는 직선과 배열에 저장된 텍스트를 fillText 를 통해 생성하는 방식으로 표현했습니다.

중앙 카트는 Image() 객체를 생성하고 src 에 카트 이미지를 지정해 create_Image() 함수로 이미지를 생성했습니다.

▲화면 좌측 메뉴바&중앙 카트 예시

c)화면 우측 상단 상태 창& 우측 하단 메뉴 아이콘

```
    var cart_status=new Polygon(950,100,50,5,Math.PI*2,'#24C4FE','#1B457E'),
            cart_status_inside=new Polygon(950,100,37,5,Math.PI*2,'#24C4FE
','#24C4FE');
        drawPolygon(cart_status);
        drawPolygon(cart_status_inside);
         context.fillStyle='white';
        context.font = '300 12pt Arial';
        context.fillText("드리프트",950,45);
        context.fillText("게이지 충전",870,70);
        context.fillText("부스터 시간",870,140);
        context.fillText("커브",1000,140);
        context.fillText("가속도",1020,90);
        context.fillText("카트 상태 표시 오각형을 클릭해",1000,170);
        context.fillText("능력치를 변경할 수 있습니다",1000,190);
        /*카트 상태창*/
```

▲화면 우측 카트 상태 창 생성 코드

화면 우측 카트 상태 창은 원본 어플을 보면 5 각형으로 이루어져 있습니다. 다각형을 생성하기 위해 정의한 Polygon()객체의 생성자로 Polygon 객체를 cart_status 와 cart_status_indside 에 지정해주고 drawPolygon() 함수를 통해 5 각형 두개를 겹치게 생성해 원본 어플의 카트 상태 창과 유사하게 표현을 하였습니다.

Cf. Polygon() 객체의 자세한 내용은 보고서 후반에 작성된 객체지향프로그래밍 파트에서 설명하겠습니다.

5 각형을 생성한 뒤 5 각형의 각마다 해당하는 text 를 fillText()를 통해 각각의 좌표를 지정해 텍스트를 생성해 주었습니다.

▲화면 우측 카트 상태 창 예시

```
var storage_down_menu_1=new Polygon(1050,450,40,6,Math.PI*2,'#496FB7','#6F86BD
'),
                storage_down_menu_2=new Polygon(965,460,25,6,Math.PI*2,'#496FB
7','#6F86BD'),
                storage_down_menu_3=new Polygon(890,460,25,6,Math.PI*2,'#496FB
7','#6F86BD'),
                storage_down_menu_4=new Polygon(815,460,25,6,Math.PI*2,'#496FB
7','#6F86BD');
            drawPolygon(storage_down_menu_1);
            drawPolygon(storage_down_menu_2);
            drawPolygon(storage_down_menu_3);
            drawPolygon(storage_down_menu_4);
        var storage_down_menu_ary = ["꾸미기","레벨업","액세서리","카트 목록"],
            storage_down_menu_count = 0;
            context.font = '400 15pt Arial';
            context.textAlign = 'center';
            context.fillStyle = 'white';
            for(var storage_down_menu_index=805;storage_down_menu_index<1080;s
torage_down_menu_index+=80){
                context.fillText(storage_down_menu_ary[storage_down_menu_count
++],storage_down_menu_index,510);
            }
             var storage_down_menu_img_1=new Image(),
                storage_down_menu_img_2=new Image(),
                storage_down_menu_img_3=new Image(),
                storage_down_menu_img_4=new Image();
            storage_down_menu_img_1.src='resource/storageIndex1.png';
            storage_down_menu_img_2.src='resource/storageIndex2.png';
            storage_down_menu_img_3.src='resource/storageIndex3.png';
            storage_down_menu_img_4.src='resource/storageIndex4.png';
            create_Image(storage_down_menu_img_1,1015,415,70,70);
            create_Image(storage_down_menu_img_2,950,445,30,30);
            create_Image(storage_down_menu_img_3,875,445,30,30);
            create_Image(storage_down_menu_img_4,800,445,30,30);
            /*창고 하단 메뉴*/
```

▲화면 우측 하단 메뉴 아이콘 생성 코드

화면 우측 하단 메뉴는 4 개의 6 각형으로 이루어져 있습니다. 이를 표현하기 위해 앞서 카트 상태창을 만들 때 사용했던 Polygon() 객체의 변의 개수를 변경하고 drawPolygon() 함수를 호출해 6 각형 4 개를 생성하고 아이콘 아래 텍스트는 for 문을 이용해 x 축을 증가시키며 배열에 있는 텍스트를 일정 간격으로 반복해서 생성하도록 코드를 작성했습니다. 아이콘 내부의 이미지의 경우 표현하기 어려운 모양들이 많아 Image() 객체를 생성하고 각 객체에 해당하는 이미지를 src 로 지정하고 create_Image() 함수로 비트맵 이미지를 생성했습니다.

▲화면 우측 하단 메뉴 아이콘 예시

d)화면 좌측 상단 화살표&카트 정보 텍스트

```
var arrow_btn=new Polygon(120,25,20,3,Math.PI/2+Math.PI,'#A9F9FA','#124882'),
                cart_info = new Image();
            context.lineWidth = 7;
            drawPolygon(arrow_btn);
            context.fillStyle='#124882';
            context.strokeStyle='#A9F9FA';
            context.lineCap = 'butt';
            context.beginPath();
            context.moveTo(130,15);
            context.lineTo(160,15);
            context.lineTo(160,35);
            context.lineTo(130,35);
            context.stroke();
            context.fill();
             context.fillStyle='white';
            context.font = '500 23pt Arial';
            context.fillText("카트",200,37);
```

▲화면 좌측 상단 화살표 생성 코드

화면 좌측 상단의 화살표는 Polygon() 객체의 생성자의 변의 개수를 3 개로 지정해 drawPolygon() 함수로 삼각형을 생성한 뒤 직선을 그리는 명령어를 이용해 'ㄷ' 모양을 만들어 화살표를 표현했습니다. 그리고 화살표 우측에 fillText() 함수로 '카트'텍스트를 생성했습니다.

▲화면 좌측 상단 화살표 예시

```
            context.font = '700 30pt Arial';
            context.fillText("사이버 버스트",350,70);
            context.font = '300 15pt Arial';
            cart_info.src = 'resource/cartInform.png';
            create_Image(cart_info,220,70,280,35);
            context.fillStyle = 'yellow';
            context.fillText("(4 일 23 시간)",535,65);
            context.font = '300 13pt Arial';
            context.fillStyle = '#A6E5FD';
            context.fillText("사이버 기술에 의해 새롭게 태어난 버스트,",380,120);
            context.fillText("강자의 귀환은 봄힐 마을에 파란을 일으킬 거예요.",40
5,140);
            /*화면 좌측 상단 카트 정보 UI*/
```

▲화면 좌측 상단 카트 정보 생성 코드

화면 좌측 상단 카트 정보는 fillText() 함수를 이용해 각각의 좌표를 지정하고 생성을 해 주었고 텍스트 중간에 위치하는 평행사변형 안에 있는 텍스트의 경우 Image() 객체를 이용해 create_Image() 함수로 표현했습니다.

▲화면 좌측 상단 카트 정보 예시

e)화면 하단 사용 중 정보

```
var using_ui = new Polygon(canvas.width/2,500,40,6,Math.PI/2,'#F2C72F','#D8681
3');
            drawPolygon(using_ui);
            context.font = '400 15pt Arial';
            context.fillStyle = 'white';
            context.lineWidth = 1;
            context.strokeStyle = '#F2C72F';
            context.fillText("사용 중",canvas.width/2,505);
            context.strokeText("사용 중",canvas.width/2,505);
            /*화면 하단 사용중 정보*/
```

▲화면 하단 사용 중 생성 코드

화면하단의 사용 중을 나타내는 텍스트와 도형은 Polygon()객체를 이용해 6 각형을 만드는데 앞선 메뉴 아이콘 6 각형과는 다르게 객체의 시작점의 각도를 변경해 옆으로 누워있는 6 각형을 생성하고 그 위에 fillText()로 텍스트를 생성했습니다.

3) create_canvas_gamestart()

create_canvas_main()함수에서 모사할 어플의 화면은 아래와 같습니다.

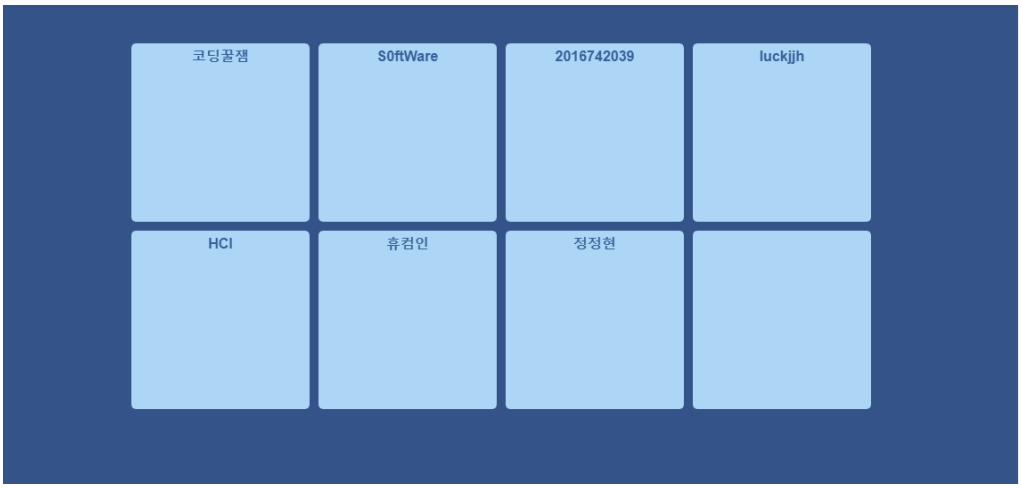
a)배경&User 들이 위치하는 Box index

```
var user_id=["코딩꿀잼","HCI","S0ftWare","휴컴인","2016742039","정정현","luckjjh"," "],
                user_id_count = 0;
            context.fillStyle='#2A548D';
            context.fillRect(0,0,canvas.width,canvas.height);
            /*게임 시작화면 배경*/
            context.lineJoin='round';
            context.lineWidth=10;
            for(var ingame_box_x=150;ingame_box_x<canvas.width-200;ingame_box_x+=210){
                for(var ingame_box_y=50;ingame_box_y<canvas.height-200;ingame_box_y+=210){
                    context.fillStyle='#A2D8F9';
                    context.strokeStyle='#A2D8F9';
                    context.strokeRect(ingame_box_x,ingame_box_y,190,190);
                    context.fillRect(ingame_box_x,ingame_box_y,190,190);
                    context.fillStyle='#2D61A0';
                    context.font='700 12pt Arial';
                    context.textAlign='center';
                    context.fillText(user_id[user_id_count++],ingame_box_x+95,ingame_box_y+15);
                }
            }/*user 들 들어가는 index*/
```

▲배경&User 들 위치하는 Box Index 생성 코드

배경은 앞서 배경을 만들었던 같은 방법으로 canvas 화면 크기를 가득 채우는 사각형을 fillRect() 함수를 이용해 생성하고 user 가 들어갈 index 는 2 중 for 문을 이용해 바깥쪽 for 문은 x 축을 증가시키고 안쪽 for 문은 y 축을 증가시켜 일정

간격을 갖는 사각형을 8 개 생성하면서 fillText() 함수를 활용해 user_id 배열에 있는 id 들을 생성했습니다.



▲배경&User 들 위치하는 Box Index 와 ID 예시

b)User 캐릭터

```
var user_img_1=new Image(),user_img_2=new Image(),user_img_3=new Image(),
            user_img_4=new Image(),user_img_5=new Image(),user_img_6=new Image
(),user_img_7=new Image();
            user_img_1.src='resource/user1.png';user_img_2.src='resource/user2
.png';
            user_img_3.src='resource/user3.png';user_img_4.src='resource/user4
.png';
            user_img_5.src='resource/user5.png';user_img_6.src='resource/user6
.png';
            user_img_7.src='resource/user7.png';
            create_Image(user_img_1,150,50,200,200);
            create_Image(user_img_2,360,40,200,200);
            create_Image(user_img_3,570,50,200,200);
            create_Image(user_img_4,770,50,205,205);
            create_Image(user_img_5,140,260,205,205);
            create_Image(user_img_6,355,260,205,205);
            create_Image(user_img_7,560,250,205,205);
```

▲User 캐릭터 생성 코드

User 의 캐릭터는 표현하기 Canvas 로 표현하기 어려운 부분이 많아 Image()객체를 이용해 각 src 를 지정하고 create_Image() 함수를 호출해 비트맵 이미지로 표현했습니다.

▲User 캐릭터 생성 예시

c)화면 상단 인터페이스&하단 인터페이스

화면 상단 인터페이스는 화살표와 방 이름, 돋보기 아이콘으로 구성되어있습니다. 화면 상단 화살표와 방 이름은 앞서 create_canvas_storage() 함수의 d) 파트의 화살표와 텍스트에서 설명을 한 코드와 동일하므로 설명을 생략하겠습니다.

```
context.fillStyle='#28AAFF';
            context.strokeStyle='#C6EEFF';
            context.fillRect(330,13,20,20);
            create_circle(343,20,4,6);
            context.lineWidth=4;
            context.beginPath();
            context.moveTo(340,23);
            context.lineTo(330,35);
            context.stroke();/*화면 상단 화살표, 방 이름, 돋보기*/
```

▲화면 상단 돋보기 아이콘 생성 예시

돋보기 아이콘은 사각형과 원, 직선을 활용해 표현했습니다. fillRect()함수로 사각형을 생성한 뒤 그 위에 create_circle() 함수를 사용해 원을 생성하고 직선을 그리는 명령어를 이용해 사선으로 직선을 그려 돋보기를 표현했습니다.

▲ 화면 상단 인터페이스 예시

화면 하단 인터페이스는 채팅 창, 각종 버튼들로 구성 되어있습니다. 하단 채팅창은 create_canvas_main() 함수에서 만든 채팅 창과 같은 방법으로 생성이 되므로 설명을 생략하겠습니다.

```
var ingame_menu = new Image(),
    ingame_btn = new Image();
ingame_menu.src = 'resource/ingameDownMenu.png';
create_Image(ingame_menu,400,455,300,70);
context.strokeStyle='#289EFE';
context.fillStyle='#289EFE';
context.lineJoin = 'round';
context.lineWidth = 20;
context.strokeRect(720,480,60,35);
context.fillRect(720,480,50,30);
ingame_btn.src = 'resource/ingameBtnImg.png';
create_Image(ingame_btn,720,470,60,50);

context.strokeRect(810,480,150,35);
context.fillRect(810,480,140,30);
context.fillStyle='#1A4F9C';
context.font='1000 20pt Arial';
context.fillText("준비 취소",885,510);/*화면 하단 버튼들*/
```

▲화면 하단 버튼 생성 예시

화면 하단 버튼 중 가장 좌측의 버튼은 Image()객체를 활용해 비트맵 이미지로 생성을 했습니다. 그리고 우측 두개 버튼은 context 의 lineJoin 을 round 로 지정해 모서리가 둥근 사각형을 만든 후 그 위에 fillText() 함수와 create_Image() 함수를 사용해 버튼을 표현했습니다.

▲ 화면 하단 인터페이스 예시

-이벤트 핸들러

Canvas 간 화면 전환을 위해 앞서 'Canvas 그리기' 파트에서 언급한 것처럼 create_canvas_xx() 함수 형태로 만들어 함수를 호출하면 Canvas 를 그리도록 했습니다.

Canvas 간 화면 전환은 두가지 방법으로 버튼 클릭과 숫자 키 클릭 두가지 방법으로 가능합니다.

1)버튼을 활용한 화면 전환

```
        #previous_btn{
            margin: 10px;
            margin-right: 940px;/*우측 버튼과 거리 조절*/
            border-top-left-radius: 5px;
            border-bottom-left-radius: 5px;
            border-top-right-radius: 5px;
            border-bottom-right-radius: 5px;/*버튼 둥글게 각도 조정*/
        }
        #next_btn{
            border-top-left-radius: 5px;
            border-bottom-left-radius: 5px;
            border-top-right-radius: 5px;
            border-bottom-right-radius: 5px;
        }
        #btn_style button{/*canvas UI 바꾸는 버튼*/
            border:1px solid rgb(20, 49, 211);
            color: rgb(20, 49, 211);
            background-color: rgb(0,0,0,0);
            font-size: 30px;
            padding: 10px;
        }
        #btn_style button:hover{/*버튼에 마우스 올라갔을때 변화*/
            color:white;
            background-color: rgb(20, 49, 211);
        }
```

▲버튼 <style> 관련 코드

Canvas 간 화면 전환을 버튼으로 하기 위해 먼저 <style>코드에 previous_btn 과 next_btn 의 스타일과 버튼 간 위치를 지정해 주었습니다.

```
<div id="btn_style">
        <button id="previous_btn" onclick="previous_btn_click();">previous
        </button>
        <button id="next_btn" onclick="next_btn_click();">next</button>
    </div>
```

▲previous&next 버튼 생성 코드

Style 을 지정해준 뒤 를 이용해 canvas 밑에 버튼 두개를 위치하게 하고 앞서 선언한 btn_style 을 div 의 id 로 지정해 버튼에 마우스가 올라갔을 경우

이미지가 바뀌게 코드를 작성했습니다. 또한 button 의 onclick 함수를 지정해 버튼이 눌렸을 경우 지정한 onclick() 함수가 호출되도록 하였습니다.

▲previous&next 버튼 생성된 예시

```
create_canvas_main();
 function previous_btn_click(){
     if(count==1){
         alert("이 canva 가 첫번째 canvas 입니다.");
     }
     else if(count==2){
         count--;
     canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
         create_canvas_main();
     }
     else if(count==3){
         count--;
     canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
         create_canvas_storage();
     }

 }
 function next_btn_click(){
     if(count==3){
         alert("이 canva 가 마지막 canvas 입니다.");
     }
     else if(count==2){
         count++;
     canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
         create_canvas_gamestart();
     }
     else if(count==1){
         count++;
     canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
         create_canvas_storage();
     }
 }
```

▲previous&next 버튼 onclick() 함수를 이용해 화면 전환을 하는 코드

먼저 default 화면이 있어야 하기 때문에 create_canvas_main() 함수를 호출해 html 파일이 실행됐을 경우 main 화면이 canvas 에 나타나게 하였습니다. 그리고

count 라는 변수를 선언한 후 1 로 초기화 시켜 count 변수를 각 페이지를 구분하는 용도로 사용했습니다. 각 버튼의 onclick 함수 내용은 버튼이 눌렸을 때 현재 count 변수의 값을 if else 문으로 확인하고 previous 버튼의 경우 canvas 를 이전 canvas 로 전환시키고 count 변수를 1 감소시켰습니다. next 버튼의 경우 canvas 를 다음 canvas 로 전환시키고 count 변수를 1 증가시키도록 코드를 작성하였습니다.

2)숫자키를 활용한 화면 전환

```
window.addEventListener("keypress",function(e){
            if(e.keyCode=='49'&&count!=1){
                count=1;
            canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
                create_canvas_main();
            }
            else if(e.keyCode=='50'&&count!=2){
                count=2;
            canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
                create_canvas_storage();
            }
            else if(e.keyCode=='51'&&count!=3){
                count=3;
            canvas.getContext("2d").clearRect(0,0,canvas.width,canvas.height);
                create_canvas_gamestart();
            }
        });
```

▲숫자키를 이용해 화면 전환을 하는 코드

키보드의 숫자 키가 눌렸을 때의 입력을 통해 화면을 전환하기 위해 먼저 window.addEventListener("keypress",function(e)) 함수를 선언해 주었습니다. 그리고 if else 문을 통해 각각의 숫자 키가 눌렸을 때 화면을 전환시키기 위해서 숫자 '1', '2', '3'의 keycode 인 '49', '50', '51'을 조건문으로 넣어주었습니다. If else 문 내부에는 기존의 canvas 를 clearRect method 로 지우고 숫자키에 해당하는 canvas 를 생성하는 함수를 호출해 canvas 간 전환이 가능하도록 코드를 작성했습니다.

-객체 지향 프로그래밍

특정 UI 요소에 대응이 되도록 두개의 객체를 정의했습니다. 개요의 요구조건 표에 나와 있듯이 두개의 객체는 Polygon() 객체와 ChattingBox()객체입니다.

1)Polygon() 객체

Polygon 객체는 다각형을 생성하기 위해 선언한 객체입니다.

```
var Polygon = function(centerX,centerY,radius,sides,startAngle,strokeStyle,fil
lStyle){
            this.x = centerX;
            this.y = centerY;
            this.radius = radius;
            this.sides = sides;
            this.startAngle = startAngle;
            this.strokeStyle = strokeStyle;
            this.fillStyle = fillStyle;
        };
```

▲Polygon 객체 생성자 함수 & 멤버변수

위 코드와 같이 Polygon 객체의 생성자 함수를 생성하고 생성자 함수의 parameter 로 값을 받아 멤버 변수를 지정하도록 했습니다. 각각의 parameter 는 centerX 와 centerY 는 다각형 중심점의 위치, radius 는 다각형의 반지름, sides 는 변의 개수, startAngle 은 다각형의 첫 변이 시작되는 위치, strokeStyle 과 fillStyle 은 외곽선과 다각형 내부의 색상을 나타냅니다.

```
            Polygon.prototype = {
            getPoints:function(){
                var points = [],
                    angle = this.startAngle || 0;
                for(var j = 0; j < this.sides; ++j){
                    points.push(new Point(this.x+ this.radius*Math.sin(angle),
                                          this.y-
this.radius*Math.cos(angle)));
                    angle+=2*Math.PI/this.sides;
                    }
                return points;
            },
            createPath:function(context){
                var points = this.getPoints();
                context.beginPath();
                context.moveTo(points[0].x,points[0].y);
```

```
            for(var k=1;k<this.sides;++k){
                context.lineTo(points[k].x,points[k].y);
            }
            context.closePath();
        },
        stroke:function(context){
            context.save();
            this.createPath(context);
            context.strokeStyle=this.strokeStyle;
            context.stroke();
            context.restore();
        },
        fill:function(context){
            context.save();
            this.createPath(context);
            context.fillStyle = this.fillStyle;
            context.fill();
            context.restore();
        },
        move:function(x,y){
            this.x=x;
            this.y=y;
        }
    };
```

▲Polygon 객체 공용 메소드와 프로토타입을 이용한 생성

Polygon 객체의 공용 메소드로는 getPoints(), createPath(context), stroke(context), fill(context), move(x,y) 메소드가 존재합니다.

getPoints() 메소드는 Polygon 객체로부터 다각형의 첫번째 변이 시작되는 각도인 startAngle 과 다각형 변의 개수인 sides 를 받아 각 꼭지점을 계산해 Point 배열에 넣고 해당 배열을 반환하는 메소드입니다.

createPath(context) 메소드는 앞선 getPoints()에서 배열을 반환 받고 context 의 lineTo() 함수를 이용해 각 꼭지점으로 이동을 하는 메소드입니다.

Stroke(context) 메소드는 createPath 로부터 꼭지점을 이동시키는데 stroke() 함수를 이용해 다각형의 외곽선을 그리는 메소드입니다.

Fill(context) 메소드는 stroke 메소드와 같은 원리로 createPath 를 이용해 fill() 함수로 다각형 내부를 지정한 색상으로 채우는 메소드입니다.

Move(x,y) 메소드는 다각형을 이동할 경우 사용되는 메소드입니다.

Polygon 객체는 총 9 번 생성되었습니다. 주로 create_canvas_storage() 함수 안에서 생성되어 해당 함수에 생성된 내역을 보여드리겠습니다.

```
var cart_status=new Polygon(950,100,50,5,Math.PI*2,'#24C4FE','#1B457E'),
                cart_status_inside=new Polygon(950,100,37,5,Math.PI*2,'#24C4FE
','#24C4FE');
            drawPolygon(cart_status);
            drawPolygon(cart_status_inside);

            canvas.onmousedown = function(e){
                if(e.x>900&&e.y>100&&e.x<1000,e.y<300&&count==2){
                    var prev_value = cart_status_inside.radius;
                    cart_status_inside.radius = Math.floor(Math.random()*47);
                    if(prev_value>cart_status_inside.radius){
                        alert('카트 능력치가 감소되었습니다.');
                    }
                    else
                        alert('카트 능력치가 증가했습니다.');
                    drawPolygon(cart_status);
                    drawPolygon(cart_status_inside);

                }
            }/*클릭시 UI 시각적 변화*/
```

▲Polygon 객체를 생성한 내역 (1)

```
var storage_down_menu_1=new Polygon(1050,450,40,6,Math.PI*2,'#496FB7','#6F86BD
'),
                storage_down_menu_2=new Polygon(965,460,25,6,Math.PI*2,'#496FB
7','#6F86BD'),
                storage_down_menu_3=new Polygon(890,460,25,6,Math.PI*2,'#496FB
7','#6F86BD'),
                storage_down_menu_4=new Polygon(815,460,25,6,Math.PI*2,'#496FB
7','#6F86BD');
            drawPolygon(storage_down_menu_1);
            drawPolygon(storage_down_menu_2);
            drawPolygon(storage_down_menu_3);
            drawPolygon(storage_down_menu_4);
```

▲Polygon 객체를 생성한 내역 (2)

```
var arrow_btn=new Polygon(120,25,20,3,Math.PI/2+Math.PI,'#A9F9FA','#124882'),
                cart_info = new Image();
            context.lineWidth = 7;
            drawPolygon(arrow_btn);
```

▲Polygon 객체를 생성한 내역 (3)

▲Polygon 객체 생성 결과 예시 이미지

(빨간색&노란색 사각형 안에 생성된 다각형들을 확인할 수 있다.)

노란색 사각형 안에 있는 Polygon 객체로 생성된 다각형의 경우 'Polygon 객체 생성 내역(1)'의 코드를 보면 해당 다각형을 마우스로 클릭하는 경우 랜덤하게 다각형의 변인 radius 의 크기를 바꿔 카트의 능력치가 감소되거나 증가될 수 있는 시각적 차이를 나타내는 요소를 추가했습니다.

2)ChattigBox() 객체

ChattingBox 객체는 create_canvas_main(1 번 화면) 와 create_canvas_gamestart (3 번 화면)함수에 있는 채팅 창에 채팅을 입력하는 경우 대화형 상호작용을 위해 선언한 객체입니다.

```
var ChattingBox = function(x,y,font,chat){/*채팅 객체 생성자 함수*/
    this.x=x;
    this.y=y;
    this.font=font;
    this.chat=chat;
};
```

▲ChattingBox 객체 생성자 함수 & 멤버변수

ChattingBox 객체의 생성자 함수는 x, y, font, chat 를 parameter 로 받습니다. Parameter x, y 는 채팅 텍스트가 생성될 위치 font 는 채팅 텍스트의 폰트 chat 은 채팅 텍스트의 내용을 받습니다.

```
ChattingBox.prototype = {
    sendChat:function(context){
        context.fillStyle = 'white';
        context.font = this.font;
        context.textAlign='center';
        context.fillText(this.chat,this.x,this.y);
```

```
        },
        removeChat:function(context){
            if(count==1){
                context.fillStyle = '#786F68';
                context.strokeStyle = 'black';
                context.fillRect(125,400,200,50);
                context.strokeRect(75,400,250,50);
            }
            else if(count==3){
                context.lineWidth=4
                context.fillStyle = '#233C65';
                context.strokeStyle = 'black';
                context.fillRect(190,470,200,50);
                context.strokeRect(140,470,250,50);
            }
        },
        responseChat:function(context){
            var inputChat=document.getElementById("chatting").value;
            if(inputChat=="안녕"||inputChat=="hi"||inputChat=="hello"
            ||inputChat=="안녕하세요"||inputChat=="ㅎㅇ"){
                this.chat = "하이하이"
                this.removeChat(context)
                this.sendChat(context)
            }
          비슷한 if else 구문이 반복되므로 아래 코드는 생략했습니다.
          }
};
```

▲ChattingBox 객체 공용 메소드와 프로토타입을 이용한 생성

ChattingBox 객체의 공용 메소드로는 sendChat, removeChat, responseChat 3 개가 존재합니다.

sendChat 메소드는 객체의 채팅 내용인 chat 멤버 변수와 채팅이 생길 위치인 멤버변수 x, y 를 받아 텍스트를 생성해주는 메소드입니다.

removeChat 메소드는 context 의 fillRect 와 strokeRect 를 이용해 현재 작성되어 있는 텍스트가 있다면 채팅 창과 같은 사각형을 생성해 텍스트를 덮어씌워 텍스트를 지우는 메소드입니다.

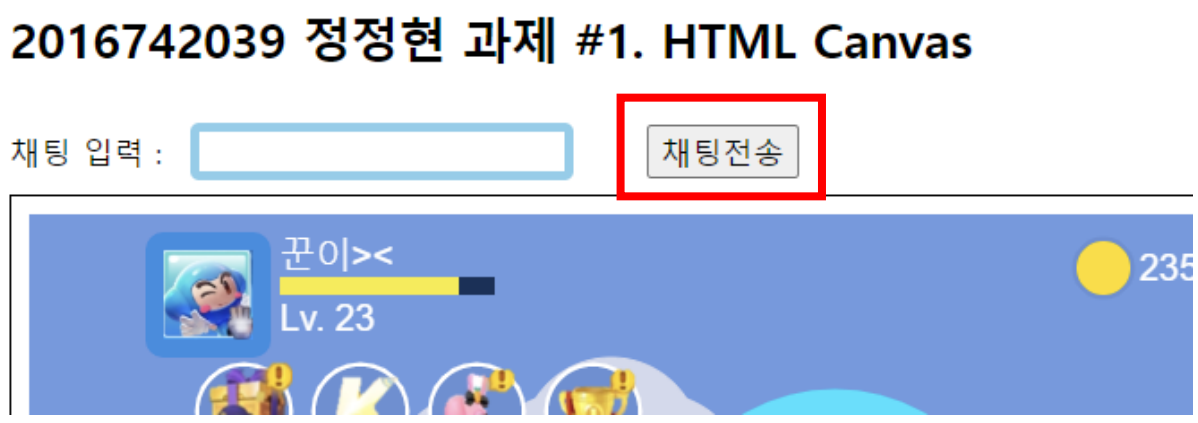
responseChat 메소드는 html 파일을 실행하게 되면 화면 상단에 텍스트를 입력할 수 있는 textBox 가 존재하는데 이곳에 텍스트가 입력되고 버튼이 눌리게 되면 전송되는 채팅에 따라 특정 반응 텍스트가 채팅창에 생성되는 메소드입니다.

```
function chat_btn_click(){
            var inputData=document.getElementById("chatting").value,
                chat_input_1 = new ChattingBox(220,430,'700 15pt Arial',inputD
ata),
                chat_input_3 = new ChattingBox(295,500,'700 15pt Arial',inputD
ata);
            if(count==1){
                chat_input_1.removeChat(context);
                chat_input_1.sendChat(context);
                sleep(2000).then(()=>chat_input_1.responseChat(context));
            }
            else if(count==2){
                alert("채팅을 지원하지 않는 canvas 입니다.")
            }
            else{
                chat_input_3.removeChat(context);
                chat_input_3.sendChat(context);
                sleep(2000).then(()=>chat_input_3.responseChat(context));
            }
        }
```

▲ChattingBox 객체를 생성한 내역

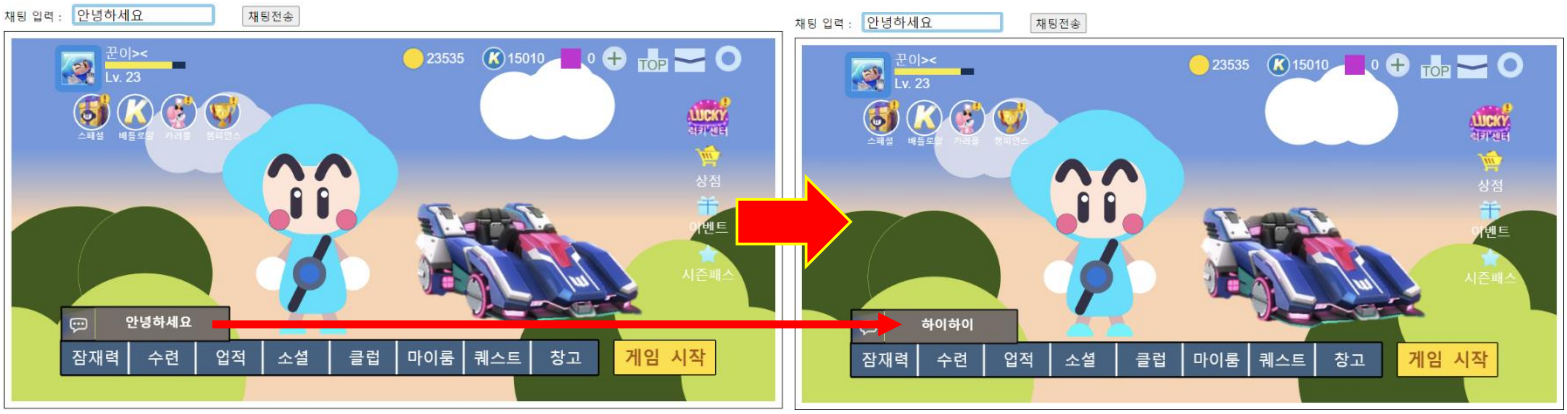
★추가 구현(대화형 상호작용)

위의 'ChattingBox 객체를 생성한 내역'을 보면 chat_btn_click() 함수안에 객체가 생성된 것을 알 수 있습니다. Chat_btn_click() 함수는 아래쪽 그림에 있는 채팅 전송 버튼에 할당된 onclick 함수입니다.



채팅 입력 텍스트 우측에 생성된 text Box 에 사용자가 텍스트를 입력하고 채팅 전송 버튼을 클릭하게 되면 chat_btn_click() 함수 내부에 있는 inputData 변수로 텍스트가 전달이 됩니다. 해당 텍스트는 ChattingBox 객체에 생성자 함수를 통해 멤버 변수인 chat 에게 전달이 됩니다. 그리고 현재 canvas 에 표현되어 있는

화면에 따라 해당하는 ChattingBox 객체의 공용 메소드인 removeChat 을 사용해 먼저 기존에 있는 텍스트를 제거하고 sendChat 메소드를 호출해 chat 에 전달된 텍스트를 출력합니다. 그 후 sleep() 함수를 호출해 2 초의 딜레이를 준 뒤 입력받은 텍스트에 대한 반응을 responseChat 메소드를 호출해 출력합니다.



▲ChattingBox 객체를 이용한 대화형 상호작용의 예시(1)

"안녕하세요"라는 텍스트 입력을 받고 canvas 채팅창에 안녕하세요가 띄워진 2 초후 "하이하이"라는 텍스트가 응답된 것을 확인할 수 있습니다.

▲ChattingBox 객체를 이용한 대화형 상호작용의 예시(2)

"게임 이름이 뭐야?"라는 텍스트 입력을 받고 canvas 채팅창에 안녕하세요가 띄워진 2 초후 "카트라이더 러쉬"라는 텍스트가 응답된 것을 확인할 수 있습니다.

사용자가 텍스트를 입력했을 때 특정 반응을 하는 텍스트들은 아래 표와 같습니다.

입력 텍스트	반응 텍스트
안녕, 안녕하세요, hi, hello, ㅎㅇ	하이하이
게임 이름이 뭐야?, 게임 이름, 이 게임 뭐야?, 게임 이름 뭐야?	카트라이더 러쉬
개발자가 누구야?, 누가 만들었어?, 개발자, 만든사람	정정현
시작할까?, 빨리 준비해, ㄹㄷㄹㄷ	ㅇㅋㅇㅋ
undefined	뭐라고 하는거야?

■논의

이번 과제는 한 어플의 화면들 중 3 개 이상을 HTML Canvas API 를 활용해 모사하는 과제였습니다. 먼저 성공적인 부분으로는 과제의 요구사항을 모두 만족하면서 처음 사용하는 2D context 를 기존에 알고 있는 Java 프로그래밍 지식과 포토샵의 이미지 크롭, 색상 코드 기능을 통해 잘 활용했다고 생각하기 때문에 굉장히 성공적이었다고 생각합니다. 또한 웹프로그래밍을 선 이수하지 않았던 저는 HTML, CSS, JavaScript 에 무지한 상태였지만 이번 과제를 통해 웹프로그래밍에 대해 조금이나마 익숙해진 졌다는 점이 있습니다. 이를 통해 앞으로의 과제나 수업, 프로젝트도 더 발전하며 진행할 수 있을 것이라 생각합니다.

실패한 부분으로는 HTML, CSS, JavaScript 를 처음 접해봐서 코드 작성을 하는 초기에는 function 이나 객체 지향 방식으로 코드를 작성하지 못해 코드가 길어졌다는 부분입니다. 그리고 대화형 상호작용을 위해 텍스트박스를 생성해 Enter key 를 누르는 경우에도 텍스트를 출력시키는 코드를 작성하려 했지만 잘 작동하지 않아 생략한 부분이 아쉬웠습니다.

향후 개선점으로는 앞서 실패한 부분에 언급한 function 이나 JavaScript 객체 지향 프로그래밍을 좀더 학습해 다음 과제나 코드를 작성할 때 효율적이고 메모리를 낭비하지 않는 방향으로 작성해야 하는 개선점이 있습니다.

-보고서 끝-