제목: 공 잡기- 클럽 리더를 위한 노트

embeds: "\*.png"

...

**#수업소개**

이 프로젝트에서 아이들은 색색의 공들을 조종기의 상응하는 부분에 맞추는 게임을 만들 것입니다.

**#수업자료**

이 프로젝트엔 Scratch 2 가 필요합니다. Scratch 2 는 http://jumpto.cc/scratch-on 에서 온라인으로 사용하거나 http://jumpto.cc/scratch-off 에서 다운받아서 오프라인으로 사용할 수 있습니다.

이 프로젝트의 완성본은 <http://scratch.mit.edu/projects/44942820/#editor> 에서 보거나 이 프로젝트의 'Download Project Materials' 링크를 클릭해서 다운로드 할 수 있으며 다음을 포함하고 있습니다:

+ CatchTheDots.sb2

또한 이 프로젝트에 필요한 외부소재가 미리 받아져 있는 버전도 사용할 수 있으며 다음 링크에서 사용 하거나 http://jumpto.cc/dots-resources 이 프로젝트의 downloadable project materials에서 받을 수 있으며 다음을 포함하고 있습니다:

+ CatchTheDotsResources.sb2

이 링크엔 또한 아이들이 프로젝트를 하는데 필요한 이미지가 들어 있는 'Project Resources' 폴더가 있습니다. 모든 아이들이 이 소재들을 가지고 시작할 수 있도록 확인해주세요.

프로젝트 소재:

+ controller.svg;

**#배움 목표**

+ 이 프로젝트는 이전에 배운 프로그래밍 기술들을 통합시키는 걸 목표로 합니다.

**#도전과제**

+ "왼쪽으로 돌기" - 조종기를 움직일 코드 만들기;

+ "더 많은 공들" - 조종기의 맞는 부분에 맞출 더 많은 색깔 공들 만들기';

+ "더 빠르게 움직이는 공들" - 변수를 사용해서 공들이 게임 중 더 빠르게 움직이게 하기;

+ "게임 개선하기!" - `score`{데이터}, `delay`{데이터} 와 `speed`{데이터} 변수에 영향을 주는 새로운 공들 만들어 넣기;

+ "게임 메뉴" - 다양한 배경과 그 배경을 고를 수 있는 버튼이 있는 메뉴 만들기.

-------------------------------------------------수업내용-------------------------------

제목: 공 잡기

난이도: Scratch 2

stylesheet: scratch

embeds: "\*.png"

materials: ["Club Leader Resources/\*", "Project Resources/\*"]

beta: true

...

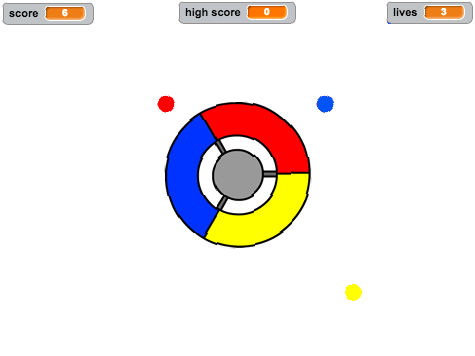
**# 소개**

이 프로젝트에선 색색별의 공들을 조종기의 알맞는 부분에 맞추는 게임을 만들 것입니다.

<div class="scratch-preview">

<iframe allowtransparency="true" width="485" height="402" src="http://scratch.mit.edu/projects/embed/44942820/?autostart=false" frameborder="0"></iframe>

</div>



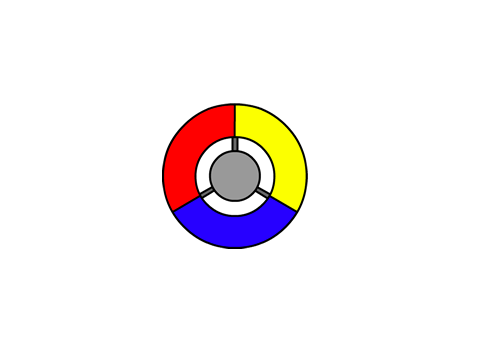
**# 1 단계: 조종기 만들기**

공을 모을 조종기를 만드는 걸로 시작합시다.

## 단계별 체크리스트

+ 새 스크래치 프로젝트를 시작하고 고양이 스프라이트를 지워서 프로젝트를 비게 만드세요. 온라인 스크래치 에디터는 http://jumpto.cc/scratch-new 에서 찾을 수 있습니다.

+ 클럽 리더에게 'Resources' 폴더를 받았다면 '스프라이트 파일 업로드하기`를 누르고 'controller.svg' 이미지를 업로드하세요. 이 스프라이트를 스테이지 중심으로 움직이세요.



이 이미지가 없다면 스스로 그리면 됩니다!

+ 오른쪽 방향키를 누르면 조종기가 오른쪽으로 돌아가게 하세요:

깃발 클릭했을 때

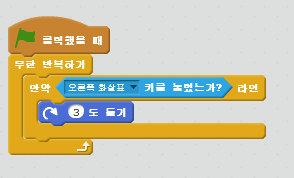
무한 반복하기

만약 <[오른쪽 화살표 v] 키를 눌렀는가?> 라면

오른쪽으로 (3) 도 돌기

end

end



+ 조종기를 시험해보세요 - 오른쪽으로 돌아가는지 확인합시다.

## 프로젝트를 저장하세요

**## 도전과제: 왼쪽으로 돌리기**

왼쪽 방향키를 누르면 조종기가 왼쪽으로 돌아가도록 할 수 있나요?

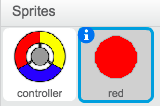
## 프로젝트를 저장하세요

**# 2 단계: 공 모으기**

조종기로 모을 공들을 더해봅시다.

## 단계별 체크리스트 { .check }

+ 새 스프라이트를 만들고 'red' 라고 이름 짓습니다. 작은 붉은색 점 모양의 스프라이트로 만드세요.



+ 이 스크립트를 'red' 스프라이트에 넣어서 몇 초에 한번씩 새 붉은 공을 복제하게 만듭시다:

깃발 클릭했을 때

(2) 초 기다리기

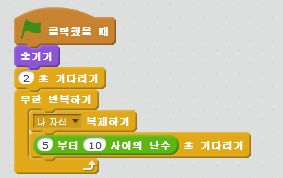
무한 반복하기

[나 자신 v] 복제하기

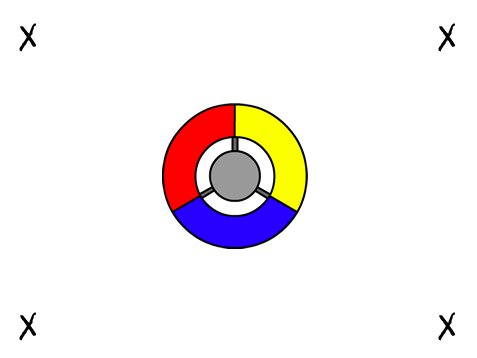
((5) 부터 (10) 사이의 난수) 초 기다리기

end

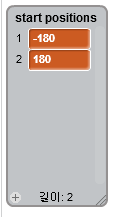
```



+ 각 복제가 만들어질 때마다 스테이지의 네 개의 모서라 중 하나에서 나오도록 만들어야 합니다.



이러려면 새 리스트 변수{데이터}를 만들고 `start positions` 이라고 이름 지은 후 `(+)`를 클릭해서 `-180` and `180`의 값을 넣읍시다.



+ 이 두 리스트를 사용해서 스테이지의 무작위 모서리를 지정하도록 만들 수 있습니다. 이 코드를 'dot'스프라이트에 더해서 각 `red` 복제품이 무작위 모서리에서 시작한 후 천천히 조종기로 다가오도록 만듭시다.

복제되었을 때

x:((random v) 번째 [start positions v] 항목) y:((random v) 번째 [start positions v] 항목) 로 이동하기

[controller v] 쪽 보기

보이기

<[controller v] 에 닿기?> 까지 반복하기

(1) 만큼 움직이기

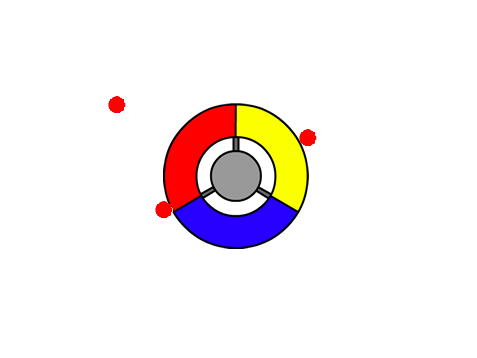
end

```



위의 코드는 x 와 y 좌표에 `-180` 이나 `180`을 지정해서 각 복제가 스테이지의 모서리 중 하나에서 시작하게 하는 기능을 가지고 있습니다.

+ 프로젝트를 시험해보세요. 붉은 공 여러 개가 각 모서리에서 나타난 후 조종기 쪽으로 천천히 움직여야 합니다.



+ 신규 변수 `lives`{데이터} 와 `score`{데이터}를 만드세요.

+ 스테이지에 게임 시작 시 `lives`{데이터} 를 3으로, `score`{데이터} 를 0으로 지정하는 코드를 넣으세요.

+ 붉은 공의 `복제 되었을 때` 코드 마지막에 이 코드를 넣어서 색깔이 맞으면 `score` 에 1이 더해지고 다르면 `lives` 에서 1이 빠지게 만드세요.

(5) 만큼 움직이기

만약 <[#FF0000] 색에 닿기?> 라면

[score v] 을(를) (1) 만큼 바꾸기

[pop v] 소리내기

아니면

[lives v] 을(를) (-1) 만큼 바꾸기

[laser1 v] 소리내기

end

이 복제본 삭제하기

```



+ 이 코드를 스테이지의 스크립트 마지막에 넣어서 플레이어의 `lives`가 0이 되면 게임이 끝나게 만드세요:

<(lives) < [1]> 까지 기다리기

[모두 v] 멈추기

```



+ 게임을 시험해서 이 코드들이 제대로 작동하는지 확인하세요.

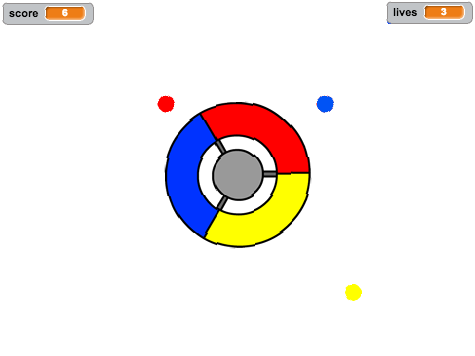
## 프로젝트를 저장하세요

**## 도전과제: 더 많은 공들**

'red' 스프라이트를 두 번 복사하고 'yellow' 와 'blue'라고 이름 지으세요.



각 스프라이트를 수정해서 (코드를 포함해서), 각 색깔의 공들이 조종기에 있는 색과 맞도록 만드세요. 프로젝트를 꼭 시험해서 점수를 얻는것과 목숨을 잃는게 올바르게 일어나는지 확인하고 게임이 너무 쉽거나 어렵지 않도록 하세요!



## 프로젝트를 저장하세요

**# 3 단계: 난이도 올리기**

플레이어가 오래 살아남을수록 공이 나타나는 사이의 간격이 줄어들게 만들어 게임의 난이도를 올립시다.

## 단계별 체크리스트

+ `delay`{데이터} 라는 새로운 변수를 만드세요.

+ 스테이지에 delay를 일단 높은 숫자로 지정한 후 천천히 줄이는 스크립트를 만들어 넣으세요.

깃발 클릭했을 때

[delay v] 에 (8) 저장하기

<(delay) = (2)> 까지 반복하기

(10) 초 기다리기

[delay v] 을(를) (-0.5) 만큼 바꾸기

end

```



게임 타이머를 만드는 방법과 매우 유사하단 점에 유의해 주세요!

+ 마지막으로 이 `delay` 변수를 붉은색, 노란색과 파란색 공들의 스크립트에 사용할 수 있습니다. 복제 하는데 무작위 초를 기다리는 코드를 지우고 `delay` 변수로 바꿔 넣으세요:

(delay) 초 기다리기

```



+ `delay` 변수를 시험해서 공을 복제하는 시간이 천천히 줄어드는지 확인하세요. 모든 색깔의 공들에게 적용되나요? `delay` 변수의 값이 줄어드는게 보이나요?

## 프로젝트를 저장하세요

**## 도전과제: 더 빠르게 움직이는 점들**

`speed`{데이터} 변수를 넣어서 공들이 처음엔 1씩 움직이다가 지속적으로 더 빠르게 움직이게 만들 수 있나요? 적용하는 방법은 위의 `delay` 변수와 매우 흡사할 것이며 그 코드를 이용하면 더 쉽게 만들 수 있습니다.

## 프로젝트를 저장하세요

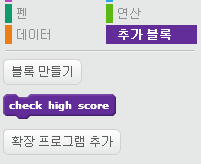
**# 4 단계: 최고 득점**

최고 득점을 저장해서 플레이어들이 얼마나 잘 하고 있는지 확인할 수 있도록 만듭시다.

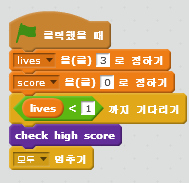
## 단계별 체크리스트

+ `high score`{데이터}라는 새 변수를 만드세요.

+ 스테이지를 클릭하고 `check high score`{추가블록} 라는 새 블록을 만드세요.



+ 게임이 끝나기 직전 부분에 새 블록을 넣으세요.



+ 이 블록에 코드를 넣어 현재 `score` 를 `만약에`{제어} 이제까지 중 가장 높은 점수라면 `high score` 로 저장하게 만드세요:

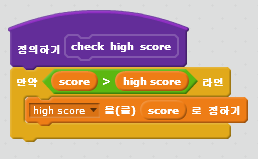
정의하기 [check high score]

만약 <(score) > (high score)> 라면

[high score v] 에 (score) 저장하기

end

```



+ 새 코드를 시험해보세요. `high score` 가 제대로 적용되는지 확인해보세요.

## 프로젝트를 저장하세요

**## 도전과제: 게임 개선하기!**

게임을 개선할 방법들을 생각해 볼 수 있나요? 예를 들자면 이런 기능을 가진 특별한 점들을 만들 수 있습니다:

+ 점수를 두배로 만들기;

+ 공들을 더 느리게 만들기;

+ 현재 스크린에 있는 모든 점들을 안보이게 하기!

## 프로젝트를 저장하세요

**## 도전과제: 게임 메뉴**

버튼이 있는 메뉴를 게임에 더해볼 수 있나요? 도움말 스크린을 더하거나 최고 득점을 표기할 스크린을 따로 만들 수도 있습니다. 도움이 필요하다면 '뇌 게임' 프로젝트를 돌아보세요.