제목: 닷지 볼 — 클럽 리더를 위한 노트

embeds: "\*.png"

...

**#수업소개:**

이 프로젝트에서 아이들은 이전에 배운 스크래치 개념과 기술들을 이용해 플랫폼 게임을 만드는 방법을 배울 것입니다.

**#수업 자료**

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/39740618/#editor> 에서 보거나 이 프로젝트의 'Download Project Materials' 링크를 클릭해서 다운로드 할 수 있으며 다음을 포함하고 있습니다:

+ Dodgeball.sb2

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

+ DodgeballResources.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 자료:

+ background.png

**#배움 목표**

+ 이 프로젝트는 이전에 배웠던 프로그래밍 기술들을 통합 시키는 걸 목표로 합니다.

**#도전과제**

+ "개선된 점프" - 추가로 `만약에`{제어} 블록을 사용하여 다중 점프 버그를 고칩니다;

+ "무작위 공" - `난수`{연산} 블록을 사용하여 공의 모양, 크기와 복제되는 간격을 다르게 합니다;

+ "더 많은 장애물" - 게임에 스프라이트 추가하기;

+ "개선된 중력" - 추가로 `만약에`{제어} 블록을 사용하여 `gravity`{데이터} 변수의 버그 고치기;

+ "더 많은 목숨" - `lives`{데이터} 변수 추가하기.

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

제목: 닷지볼

난이도: Scratch 2

stylesheet: scratch

embeds: "\*.png"

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

...

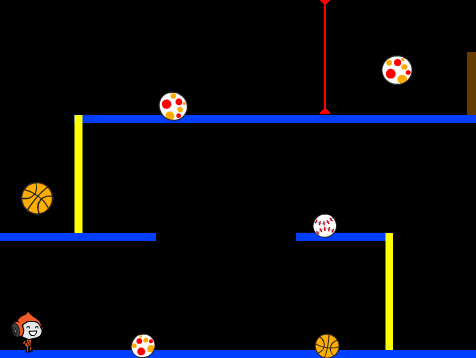
**# 소개**

이 프로젝트에선 공을 피해가며 난이도의 마지막 부분까지 가야하는 플랫폼 게임을 만들어 볼 것입니다.

<div class="scratch-preview">

<iframe allowtransparency="true" width="485" height="402" src="http://scratch.mit.edu/projects/embed/39740618/?autostart=false" frameborder="0"></iframe>

</div>



**# 1 단계: 캐릭터 움직임**

왼쪽 오른쪽으로 움직이고 봉을 타고 올라갈 수 있는 캐릭터를 만드는 것으로 시작합시다.

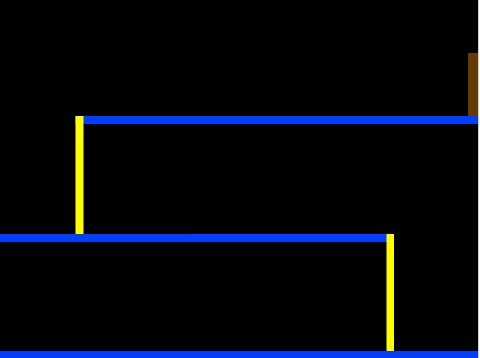
## 단계별 체크리스트

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

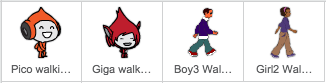
+ 이 프로젝트엔 필요하게 될 배경 이미지가 들어 있는 'Project Resources' 폴더가 필요합니다. 이 폴더를 찾아 두고 없다면 클럽 리더에게 꼭 받도록 하세요.



+ 'background.png' 이미지를 스테이지 배경으로 넣거나 스스로 배경을 그리세요! 직접 그린다면 봉과 바닥을 다른 색깔로 칠하도록 확인하고 플레이어가 다다라야 하는 문(혹은 비슷한 것)이 있도록 하세요. 프로젝트는 이런 모습이어야 합니다:



+ 자신의 캐릭터가 될 스프라이트를 더하세요. 스프라이트는 여러가지 모양이 있어서 걸어가는 것처럼 보이게 할 수 있는게 더 좋습니다.



+ 화살표 키를 사용해서 캐릭터를 움직일 수 있도록 만듭시다. 플레이어가 오른쪽 화살표 키를 누르면 오른쪽으로 조금 움직인 후 다음 모양으로 바뀌도록 해야합니다:

깃발 클릭했을 때

무한 반복하기

만약 <[오른쪽 화살표 v] 키를 눌렀는가?> 라면

(90 v) 도 방향 보기

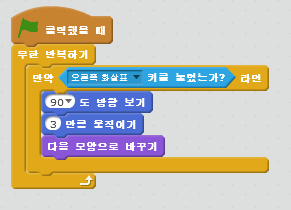
(3) 만큼 움직이기

다음 모양으로 바꾸기

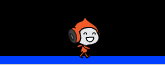
end

end

```



+ 깃발을 클릭하고 오른쪽 화살표 키를 눌러서 캐릭터를 시험해보세요. 오른쪽으로 움직이나요? 걷는 것처럼 보이나요?



+ 캐릭터를 왼쪽으로 움직이기 위해 또 하나의 `만약에`{제어} 블록이 `무한 반복하기`{제어} 루프 안에 필요합니다. 꼭 새 코드가 작동하는지 확인하세요!

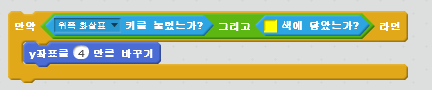
+ 봉을 타고 올라가려면 캐릭터가 알맞는 색에 닿았을 때 위쪽 화살표 키를 누르면 조금 올라가게 만들어야 합니다. 이 코드를 캐릭터의 `무한반복하기`{제어} 루프 안에 넣으세요:

만약 <<[위쪽 화살표 v] 키를 눌렀는가?> 그리고 <[#FFFF00] 색에 닿기?>> 라면

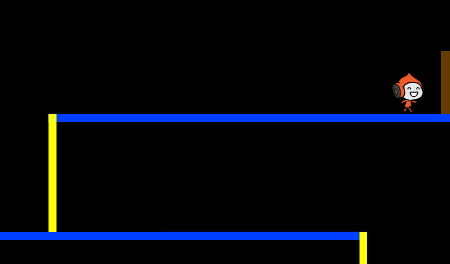
y좌표를 (4) 만큼 바꾸기

end

```



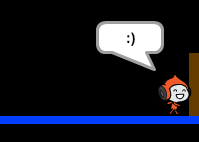
+ 캐릭터를 시험해보세요 - 노란 봉을 타고 올라가서 난이도 마지막에 다다를 수 있나요?



## 프로젝트를 저장하세요

**## 도전과제: 스테이지 끝마치기**

캐릭터에 코드를 더해서 `만약에`{제어} 갈색 문에 다다르면 무언가 말하게 할 수 있나요?



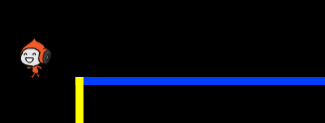
## 프로젝트를 저장하세요

**# 2 단계: 중력과 점프**

중력을 넣고 점프가 가능하게 만들어 캐릭터가 조금 더 현실적으로 움직일 수 있게 만듭시다.

## 단계별 체크리스트

+ 지금은 캐릭터가 플랫폼에서 나와 공중 위를 걸을 수도 있습니다. 플랫폼에서 걸어 나와 어떻게 되는지 확인해보세요.



+ 이 문제를 고치기 위해 게임에 중력을 더합시다. 새 변수 `gravity` 를 만드세요. 원하신다면 스테이지에서 이 변수가 안보이게 해 둘 수 있습니다.



+ 중력을 음수로 두는 이 코드블록을 더하고 이 블록을 사용해서 캐릭터의 y 좌표를 계속 바꾸게 만드세요.

깃발 클릭했을 때

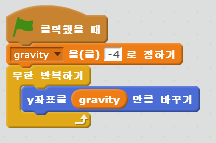
[gravity v] 에 [-4] 저장하기

무한 반복하기

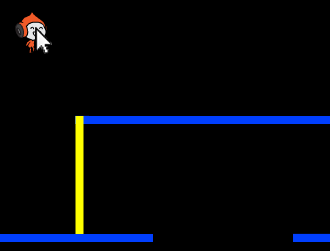
y좌표를 (gravity) 만큼 바꾸기

end

```



+ 깃발을 클릭하고 캐릭터를 스테이지 최상단으로 드래그 해 올려보세요. 어떻게 되나요? 중력이 생각한 것처럼 작동하나요?



+ 중력이 캐릭터를 플랫폼이나 봉을 뚫고 떨어지게 만들면 안됩니다! 코드에 `만약에` 블록을 더해서 캐릭터가 떠 있을 때만 중력이 작동하도록 만드세요. 중력 코드는 이제 이렇게 돼야합니다:

깃발 클릭했을 때

[gravity v] 에 [-4] 저장하기

무한 반복하기

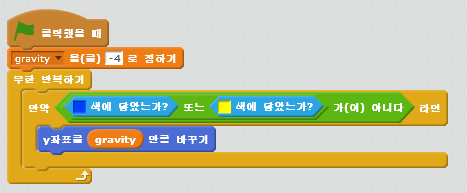
만약 <<<[#0000FF] 색에 닿기?> 또는 <[#FFFF00] 색에 닿기?>> 가(이) 아니다> 라면

y좌표를 (gravity) 만큼 바꾸기

end

end

```



+ 중력을 다시 시험해보세요. 캐릭터가 플랫폼이나 봉에 있으면 멈추나요? 플랫폼 끄트머리에서 떨어져서 아래 단계로 내려갈 수 있나요?



+ 또한 플레이어가 스페이스 바를 누르면 점프하게 만듭시다. 한가지 간단한 방법은 이 코드를 사용해서 캐릭터를 위로 몇 번 올라가게 만드는 것입니다:

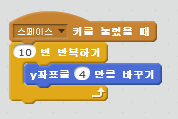
[스페이스 v] 키 눌릴 때

(10) 번 반복하기

y좌표를 (4) 만큼 바꾸기

end

```



중력이 계속해서 캐릭터를 4 픽셀씩 밀어 내리고 있으니 `y좌표를 (4)만큼 바꾸기` 블록에서 4보다 더 높은 숫자를 지정해야 합니다. 캐릭터가 점프하는 높이가 맘에 들때까지 바꿔보세요.

+ 이 코드를 시험해보면 작동은 하지만 움직임이 부드럽지 않은 걸 볼 수 있습니다. 점프 하는 걸 더 부드럽게 보이게 하려면 캐릭터가 계속해서 더 조금씩 움직이다가 아예 위로 움직이지 않을 때까지 숫자를 낮춰야 합니다.

+ 이렇게 하려면 `jump height` 이라는 변수를 만드세요. 이 변수도 원하신다면 스테이지에서 가려 둬도 됩니다.

+ 캐릭터에 더해 둔 점프 코드를 지우고 이 코드로 바꿔 넣으세요:

[스페이스 v] 키 눌릴 때

[jump height v] 에 [8] 저장하기

<(jump height) = [0]> 까지 반복하기

y좌표를 (jump height) 만큼 바꾸기

[jump height v] 을(를) (-0.5) 만큼 바꾸기

end

```



이 코드는 캐릭터를 8픽슬 올리고 그 후엔 7.5 픽슬, 7픽슬로 점프가 끝날 때까지 숫자를 낮춰갑니다. 이러면 점프하는게 훨씬 부드러워 보입니다.

+ `jump height` 변수의 처음 값을 캐릭터가 뛰는 높이에 만족할 때까지 바꿔주세요.

## 프로젝트를 저장하세요

**## 도전과제: 개선된 점프**

지금 캐릭터는 떠있어도 스페이스바를 누르면 계속 점프할 수 있습니다. 그냥 스페이스바를 누르고 만 있어도 확인할 수 있죠. 이 문제를 고쳐서 캐릭터가 `만약에` 파란색 플랫폼에 닿아 있을 때만 점프하게 만들 수 있나요?

## 프로젝트를 저장하세요

**# 3 단계: 공 피하기**

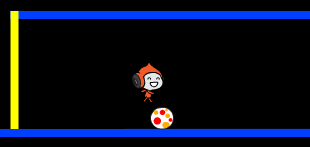
이제 캐릭터가 움직일 수 있으니 피해야 할 공들을 더해 넣어 봅시다.

## 단계별 체크리스트

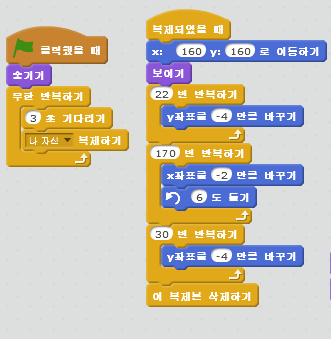
+ 새 공 스프라이트를 만드세요. 어떤 공이라도 괜찮습니다.



+ 공의 크기를 조정해서 캐릭터가 뛰어 넘을 수 있게 해주세요. 공을 뛰어 넘을 수 있는지 확인해 주세요.

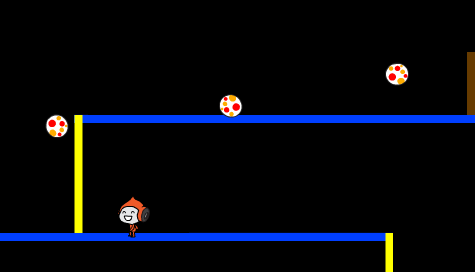


+ 공에 이 코드를 더해주세요:

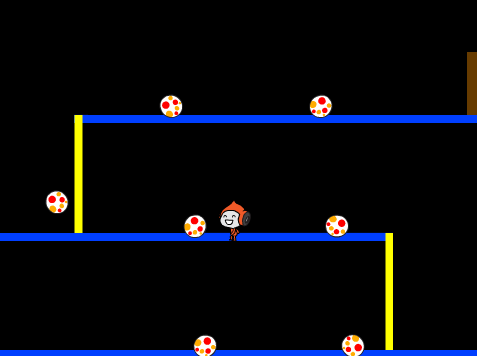


이 코드는 공을 3초마다 복제합니다. 각 복제품은 맨 위의 플랫폼을 따라 움직입니다.

+ 깃발을 클릭해서 시험해보세요.



+ 공 스프라이트에 코드를 더 넣어서 플랫폼 세 개를 모두 질러가게 만드세요.



+ 마지막으로 캐릭터가 공에 맞았을 때의 코드가 필요합니다! 이 코드를 공 스프라이트에 더해주세요:

복제되었을 때

무한 반복하기

만약 <[Pico walking v] 에 닿기?> 라면

[hit v] 방송하기

end

end

```



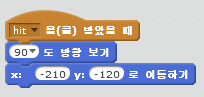
+ 또한 캐릭터도 맞았을 때 시작점으로 돌아가게 만드는 코드를 넣어야 합니다:

[hit v] 수신할 때

(90 v) 도 방향 보기

x:(-210) y:(-120) 로 이동하기

```



+ 게임을 시험해서 캐릭터가 공에 맞으면 처음으로 돌아가는지 확인해보세요.

## 프로젝트를 저장해주세요

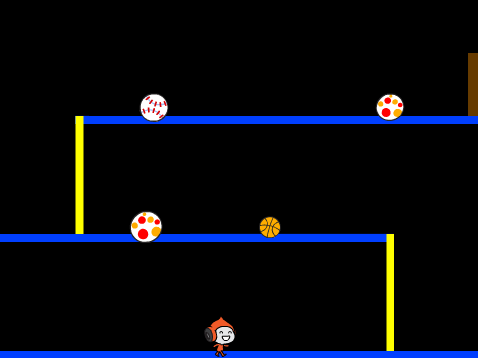
**## 도전과제: 무작위 공**

지금 캐릭터가 피해야 할 공은 전부 똑같이 생겼고 3초에 한번씩 규칙적으로 나옵니다. 이 부분을 개선해서 공이:

+ 똑같아 보이지 않게 할 수 있나요?

+ 무작위 시간 후에 생성되게 만들 수 있나요?

+ 무작위 크기로 나오게 만들 수 있나요?



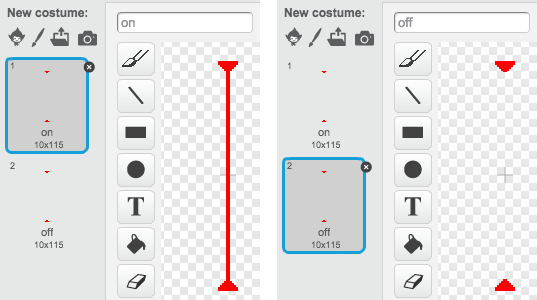
## 프로젝트를 저장하세요

**# 4 단계: 레이저!**

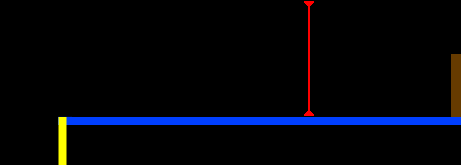
레이저를 더해서 이 게임의 난이도를 올려봅시다!

## 단계별 체크리스트

+ `Laser`라는 스프라이트를 더해주세요. 'on' 과 'off'라는 모양이 있어야 합니다.



+ 레이저를 두 플랫폼 사이 중 아무 곳이나 골라서 배치하세요.



+ 레이저에 코드를 더해 두 모양 사이로 바꾸게 만드세요.

깃발 클릭했을 때

무한 반복하기

모양을 [켜기 v] (으)로 바꾸기

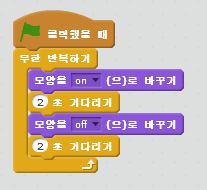
(2) 초 기다리기

모양을 [끄기 v] (으)로 바꾸기

(2) 초 기다리기

end

```



원하신다면 모양이 바뀌기 전에 `난수`{연산} 의 시간동안 `기다리기`{제어} 하게 할 수도 있습니다.

+ 마지막으로 레이저에 코드를 더해 레이저가 캐릭터에 닿으면 'hit' 메시지가 방송되게 만드세요. 이 코드는 공 스프라이트에 더했던 코드와 같습니다.

캐릭터엔 코드를 더할 필요가 없습니다- 이미 맞으면 뭘 할지 알고 있으니까요!

+ 게임을 시험해서 레이저를 피할 수 있나 보세요. 레이저가 너무 쉽거나 어렵다면 `기다리기` 시간을 바꿔보세요.

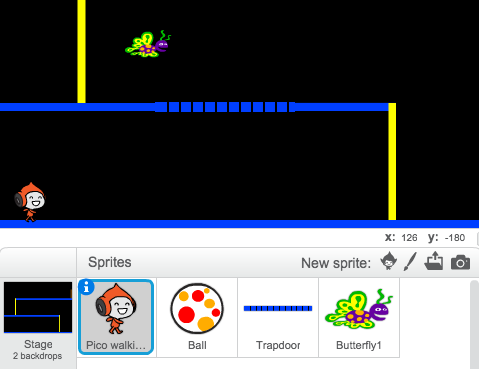
**## 도전과제: 더 많은 장애물**

아직도 게임이 너무 쉽다고 생각되면 장애물을 더 추가할 수 있습니다. 무엇을 더하던 괜찮지만 몇가지 아이디어들 입니다:

+ 날아다니는 살인 나비;

+ 나타났다가 사라지는 플랫폼r;

+ 피해야 하는 떨어지는 테니스 공들.



또한 배경을 추가하여 캐릭터가 갈색 문에 닿으면 다음 난이도로 넘어가게 할 수도 있습니다.

만약 <[#714300] 색에 닿았는가?> 라면

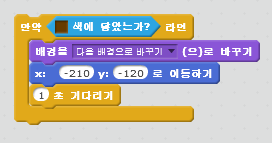
배경을 [다음 배경으로 바꾸기 v] (으)로 바꾸기

x:(-210) y:(-120) 로 이동하기

(1) 초 기다리기

end

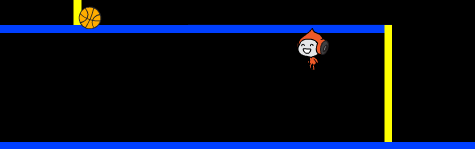
```



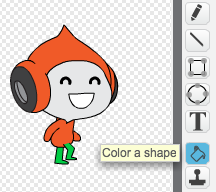
## 프로젝트를 저장하세요

**## 도전과제: 개선된 중력**

게임엔 또 하나의 작은 버그가 남아있습니다: 만약 캐릭터의 어떤 부분이라도 파란 플랫폼에 닿아 있다면, 심지어 그게 머리라도, 중력이 작동하지 않습니다! 이 문제는 봉을 거의 다 올라간 후 왼쪽으로 움직이면 알 수 있습니다.



이 문제를 해결 할 수 있겠 나요? 이걸 해결하려면 캐릭터에게 (모든 모양에서) 다른 색의 바지를 입혀야 합니다



...그리고 이 코드를:

[#0000FF] 색에 닿았는가?

```



이 코드로 바꿔 넣어야 합니다:

[#00FF00] 색이 [#0000FF] 색에 닿았는가?

```



꼭 문제가 해결됐는가 확인 해 보세요!

## 프로젝트를 저장하세요

**## 도전과제: 더 많은 목숨**

플레이어가 맞을 때마다 시작으로 돌아가는 대신 3개의 `lives`{데이터} 를 줄 수 있나요? 게임은 이렇게 작동할 수도 있습니다:

+ 플레이어는 목숨 3개로 시작합니다;

+ 플레이어가 맞을 때마다 시작으로 돌아가고 목숨이 하나 줄어듭니다;

+ 목숨이 남지 않았다면 게임이 끝납니다.

## 프로젝트를 저장하세요