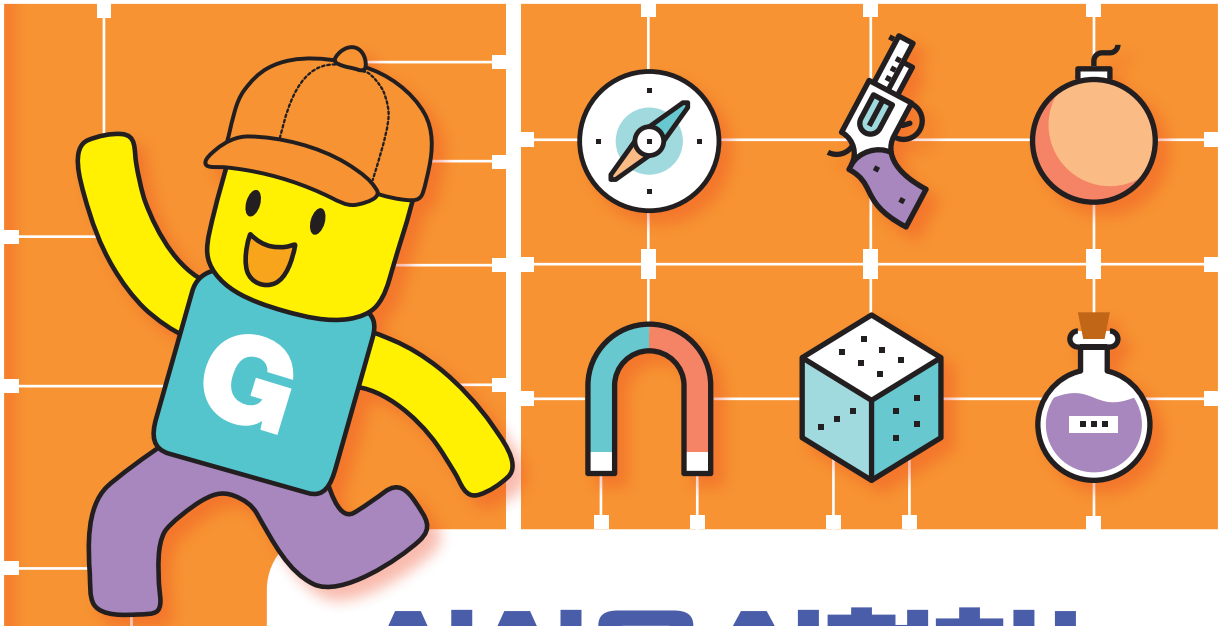
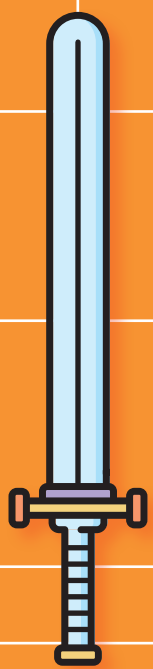
따라만 하면 손 안에 펼쳐지는 메타버스! [무료 특별판] 극한 생존 게임

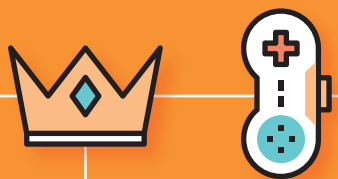


상상을 실현하는 로블록스 게임 만들기

강태훈, 장준하, D.LAB 지음



ROBLOX
UNOFFICIAL

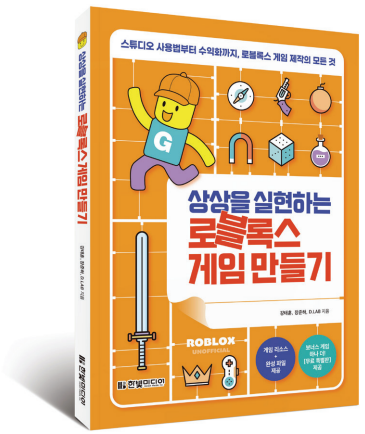


무료
특별판

한빛미디어 Hanbit Media, Inc.

무료
특별판

상상을 실현하는 로블록스 게임 만들기



극한의 환경, 뜨거운 사막에서 생수를 찾아 살아남아라!
[무료 특별판] 극한 생존 게임 만들기를 전자책으로 제공합니다.

상상을 실현하는 로블록스 게임 만들기 무료 특별판

스튜디오 사용법부터 수익화까지, 로블록스 게임 제작의 모든 것

초판 1쇄 발행 2023년 8월 23일

지은이 강태훈, 장준하, D.LAB / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 서대문구 연희로2길 62 한빛미디어(주) IT출판1부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제25100-2017-000058호
ISBN 979-11-6921-126-0 93000

총괄 배윤미 / **책임편집** 이미향 / **기획 · 편집** 박새미
디자인 윤혜원 / **일러스트** 이진숙 / **전산편집** 김현미
영업 김형진, 장경환, 조유미 / **마케팅** 박상용, 한종진, 이행은, 김선아, 고광일, 성화정, 김한솔 / **제작** 박성우, 김정우

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오. 잘못된 책은 구입하신 서점에서 교환해 드립니다. 책값은 뒤표지에 표시되어 있습니다.

한빛미디어 홈페이지 www.hanbit.co.kr / 이메일 ask@hanbit.co.kr
자료실 www.hanbit.co.kr/src/11126

Published by HANBIT Media, Inc. Printed in Korea

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(writer@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

무료
특별판

상상을 실현하는 로블록스 게임 만들기

강태훈, 장준하, D.LAB 지음

한빛미디어
Hanbit Media, Inc.

목차

부록 극한 생존 게임

부록

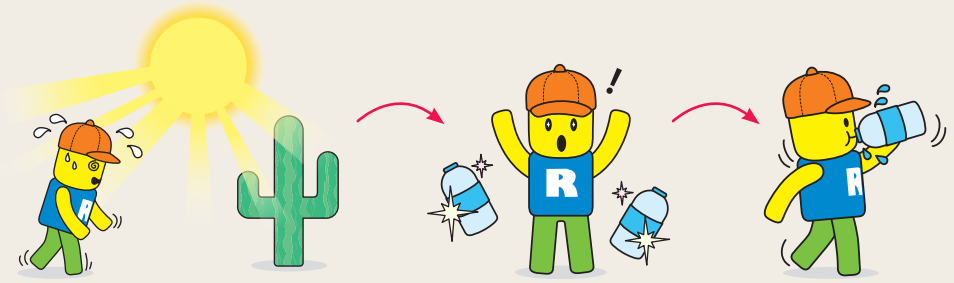
극한 생존 게임

A-1 게임 설계하기

어느 날 갑자기 사막 한복판에 뚝 떨어지게 된다면 어떻게 될까요? 가도 가도 끝이 보이지 않는 모래 언덕 앞에서 숨이 막히고, 목이 말라 쓰러질 것 같습니다. 뜨거운 사막에 숨겨져 있는 생수를 찾아 끝까지 살아남아야 합니다. 먼저 게임의 방법과 규칙을 정하고 어떻게 맵을 디자인할 것인지 생각해 봅시다.

게임 기획하기

극한의 환경, 뜨거운 사막에서 생수를 찾아 살아남아라!



게임 방법 및 규칙

- 게임을 시작하면 가만히 있어도 플레이어의 체력이 계속해서 떨어지며, 체력이 0으로 바닥나면 게임이 종료됩니다.
- 체력을 회복하려면 사막 곳곳에서 무작위로 나타나는 생수를 찾아 마셔야 합니다.
- 플레이어의 체력은 생수병을 찾아 터치하는 순간 100으로 회복됩니다.
- 제한 시간 동안 떨어지는 체력을 회복하며 끝까지 살아남아야 합니다.

제한 시간: 90초

게임 제작 순서 생각하기

다음 로드맵을 보면서 무엇을, 어떤 순서로 만들지 머릿속으로 먼저 그려 봅니다.

❶ 프로젝트 만들기

프로젝트 파일

❷ 빌드하기

사막 맵 | 선인장 모델

생수병 모델 | 투명 프레임

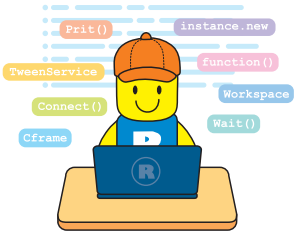
생수병 위치 지정

❸ UI, 이펙트, 사운드 추가하기

게임 규칙 UI | 라이팅 이펙트

파티클 이펙트 | 배경 음악과 효과 사운드

❹ 스크립트 작성하기



생수병 스크립트

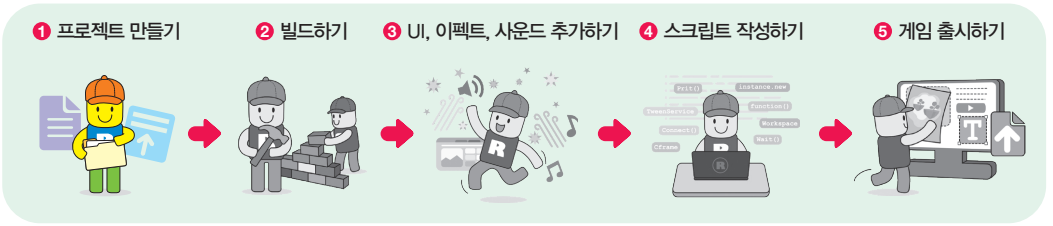
체력 스크립트

GUI 스크립트

❺ 게임 출시하기

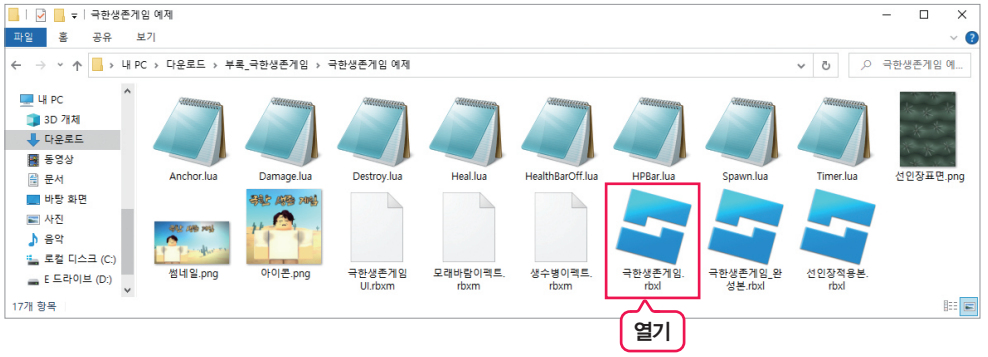
게임 출시

A-2 프로젝트 만들기

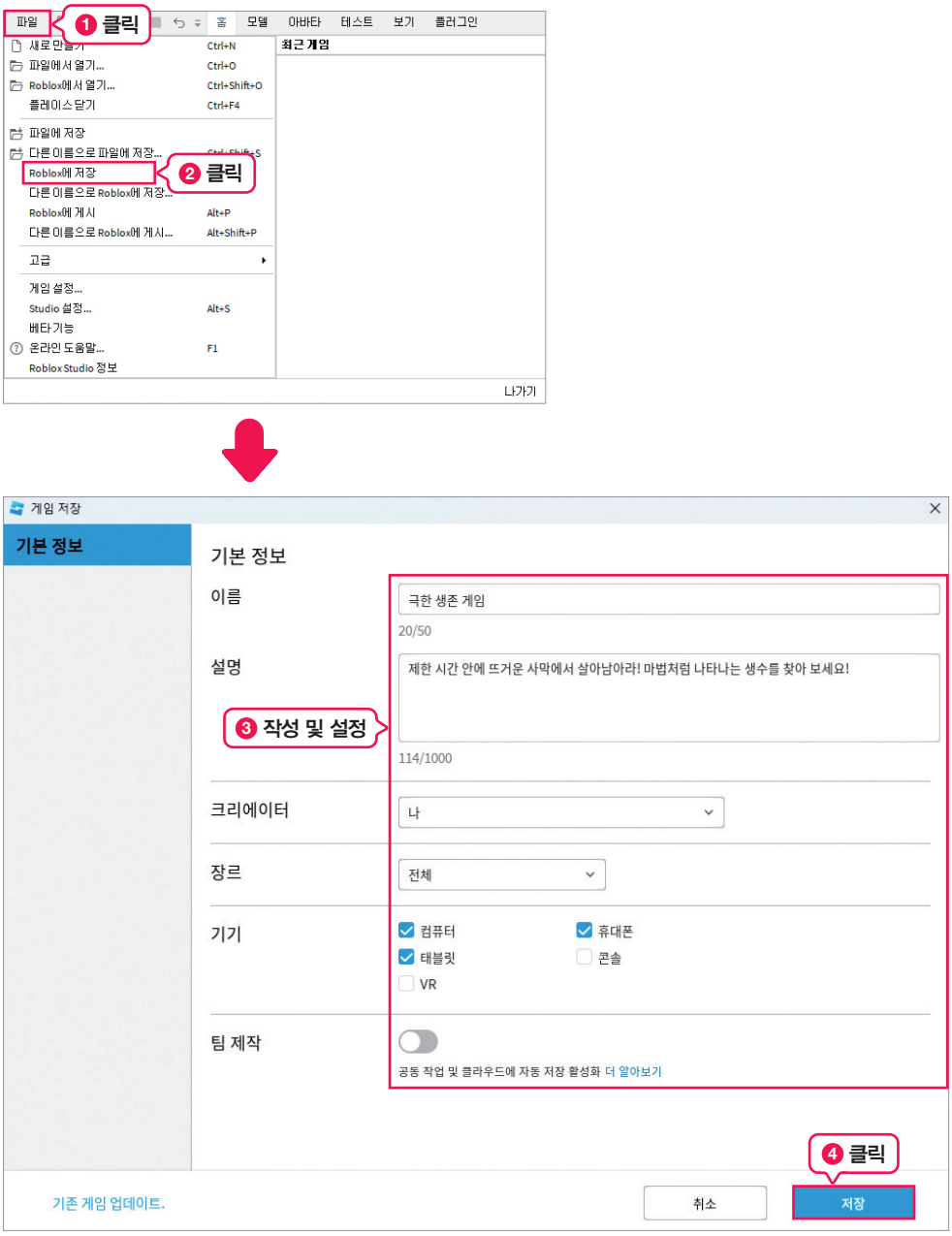


프로젝트 파일을 만들어 로블록스 스튜디오에 저장해야 합니다. 3~4장과 동일하게 예제 소스에 있는 파일을 활용해 프로젝트를 저장하겠습니다.

01 예제 소스의 [부록] 폴더에 있는 '극한생존게임.rbxl' 파일을 더블 클릭해 엽니다.

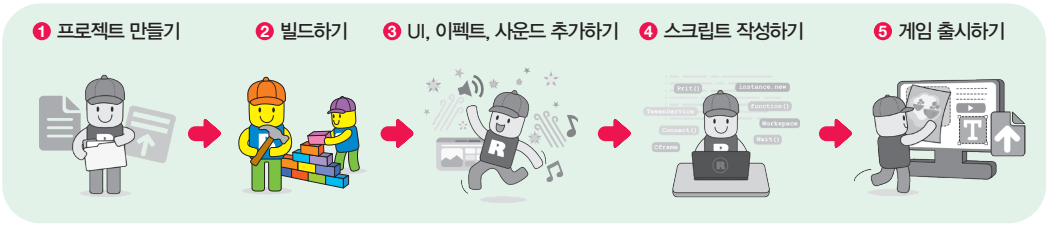


02 로블록스 스튜디오의 상단 메뉴 바에서 [파일] – [Roblox에 저장]을 클릭합니다. 게임의 [이름]을 '극한 생존 게임', [설명]을 '제한 시간 안에 뜨거운 사막에서 살아남아라! 마법처럼 나타나는 생수를 찾아 보세요!'라고 작성하고 [저장] 버튼을 클릭합니다.



03 저장을 완료하고 새로운 프로젝트가 자동으로 열리는 것을 확인합니다. 기타 세부적인 설정은 게임을 완성하고 나서 살펴보겠습니다.

A-3 빌드하기



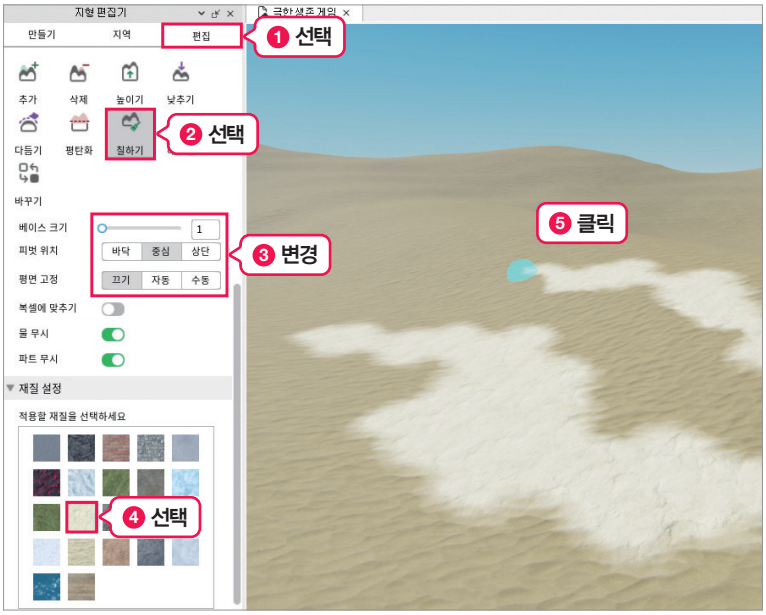
태양이 내리쬐는 사막의 모습을 떠올려 보겠습니다. 모래바람이 몰아치는 언덕들 사이로 크고 작은 선인장들이 보입니다. 구름 한 점 없는 하늘에는 뜨거운 태양이 사막을 달구고 있습니다. 아이디어가 될 만한 사막 사진들을 검색해 참고하면서 '극한 생존 게임'에 필요한 모델들을 하나씩 만들어 보겠습니다.

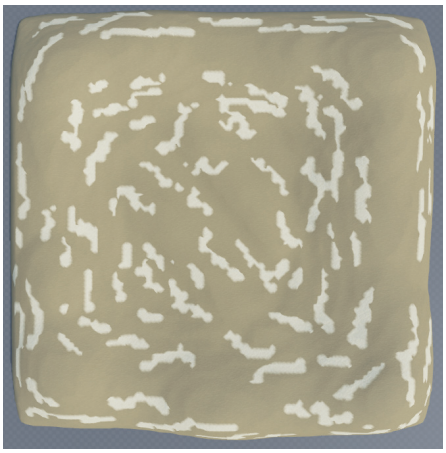
'극한생존게임.rbxl' 예제 소스를 열면 온통 모래로만 뒤덮여 있는 사막 맵을 확인할 수 있습니다. 다양한 모양과 크기의 선인장과 모래 언덕으로 사막 맵을 꾸미고, 맵의 이동 범위와 아이템 위치를 지정해 보겠습니다.

STEP 1 사막 맵 만들기

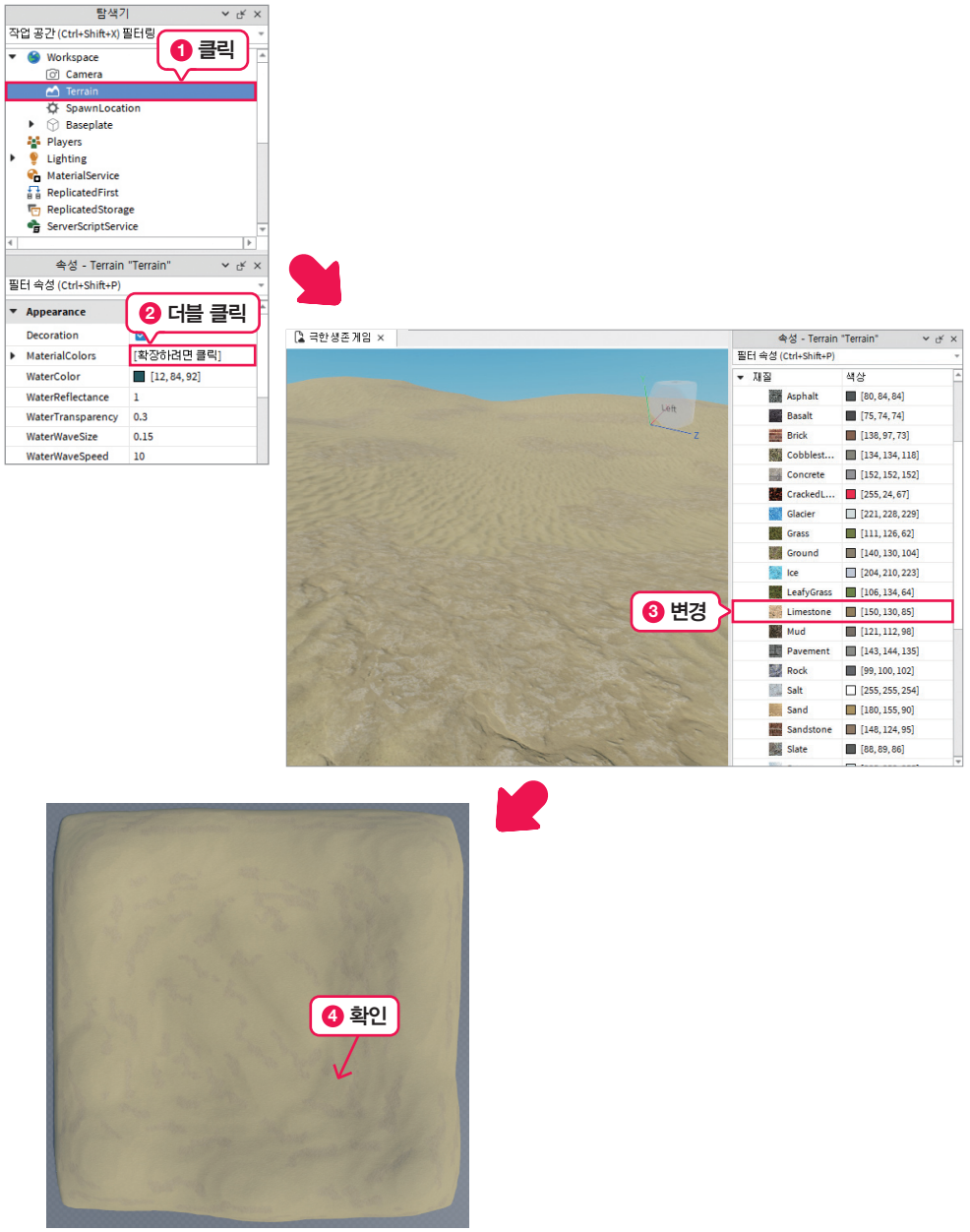
가장 먼저 [지형 편집기]를 이용해 모래 사이사이에 음영을 넣고, 선인장 모델을 추가해 실감 나는 사막의 느낌을 살려 보겠습니다.

01 모래 사막의 움푹 패인 곳에 석회암이 드러난 것처럼 음영 색을 칠해 보겠습니다. [지형 편집기] 창의 [편집] 탭에서 [칠하기] 툴을 선택한 다음, [베이스 크기]를 '1', [평면 고정]을 [끄기]로 변경하고, [재질 설정]을 '석회암' 재질로 선택합니다. 마우스로 뷰포트의 모래 지형 이곳 저곳을 클릭해 칠하면서, 특히 바닥이 좀 더 낮게 패인 지형에 부분적으로 칠합니다.

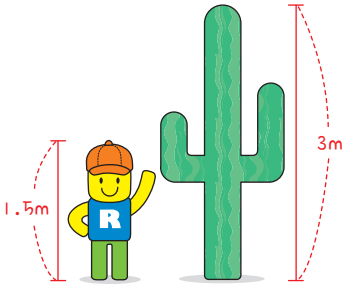


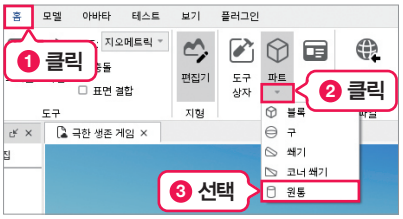


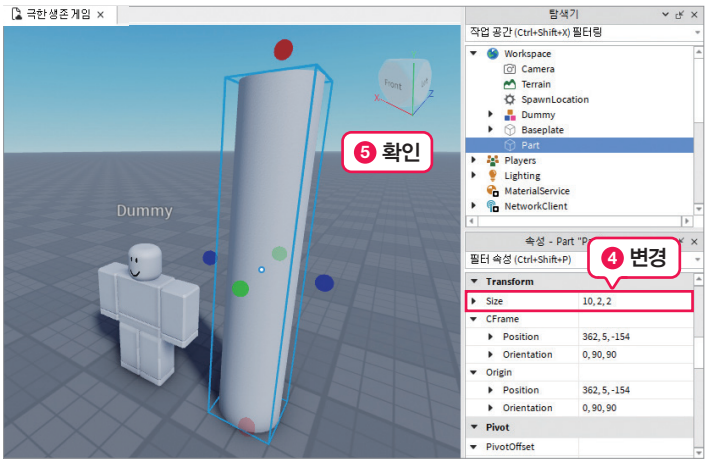
02 석회암 음영이 모래보다 밝아 어색하게 튀는 느낌이 듭니다. 석회암의 재질은 그대로 유지하고, 색상을 조금 더 모래와 어울리게 바꿔 봅시다. [탐색기] 창에 있는 [Workspace] – [Terrain(지형)]을 선택하고, [속성] 창에서 [MaterialColors]의 [확장하려면 클릭]을 더블 클릭해 재질별 기본 색상 목록을 확인합니다. [Limestone(석회암)]의 색상 값을 '150, 130, 85'로 변경합니다.



03 사막에는 실제로 사람의 키를 훌쩍 뛰어넘는 선인장이 많다고 합니다. 사막 지형 바깥에 있는 평평한 곳에서 로블록스 캐릭터보다 더 큰 선인장을 만들어 보겠습니다. [홈] – [파트]에서 [원통] 파트 하나를 생성해 수직으로 세우고, [속성] 창의 [Transform] – [Size]를 '10, 2, 2'로 설정합니다.



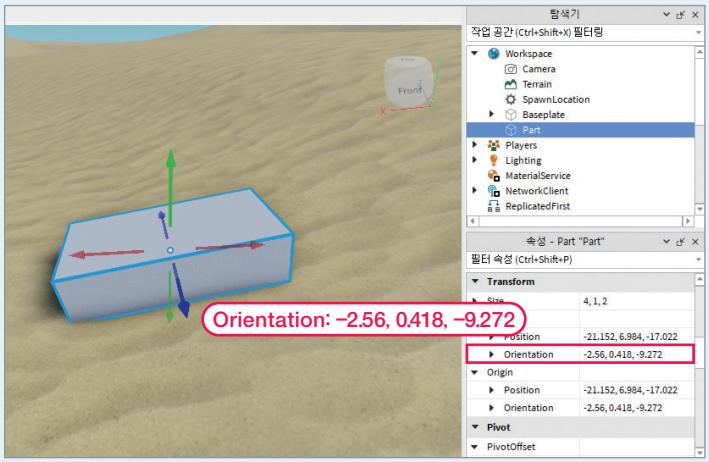




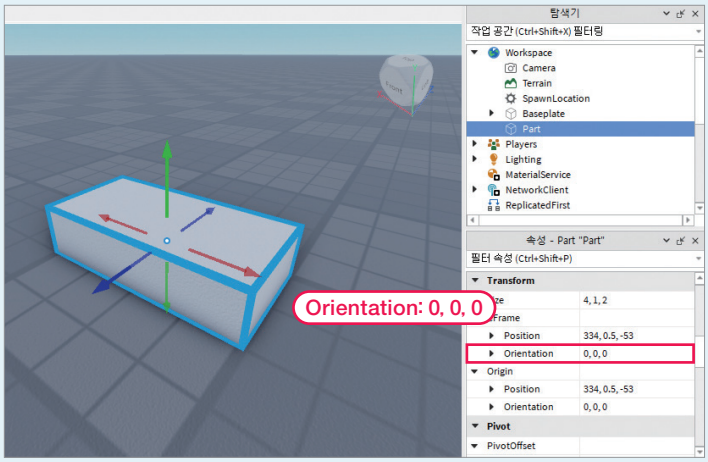
NOTE 파트를 수직으로 세울 때는 앞에서 배운 것처럼 [회전] 툴을 이용할 수도 있고, 키보드의 단축키 Ctrl + R과 Ctrl + T를 눌러 세울 수도 있습니다.

여기서 잠깐 평평한 곳에서 선인장 파트를 생성하는 이유

사막 지형 바깥의 평평한 바닥에서 선인장 파트를 만드는 이유가 있습니다. 모래로 표현되어 있는 사막 지형의 표면이 울퉁불퉁하기 때문에, 그 위에 파트를 생성하면 파트의 방향(Orientation) 속성이 지형 표면에 따라 기울어진 값으로 설정됩니다. 평평하지 않은 곳에서 만든 건물이나 물체는 어느 한 곳으로 비스듬하게 기울어질 수도 있으므로 완벽하게 평평한 곳에서 파트를 생성하는 것이 가장 좋습니다.

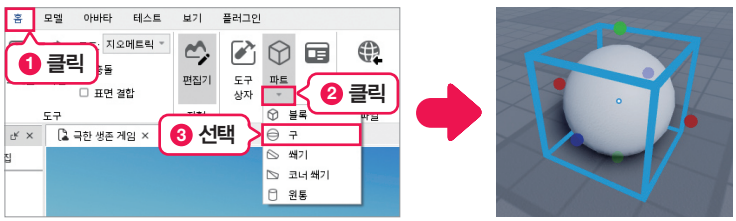


경사가 있는 지형에서 만든 파트의 Orientation 값

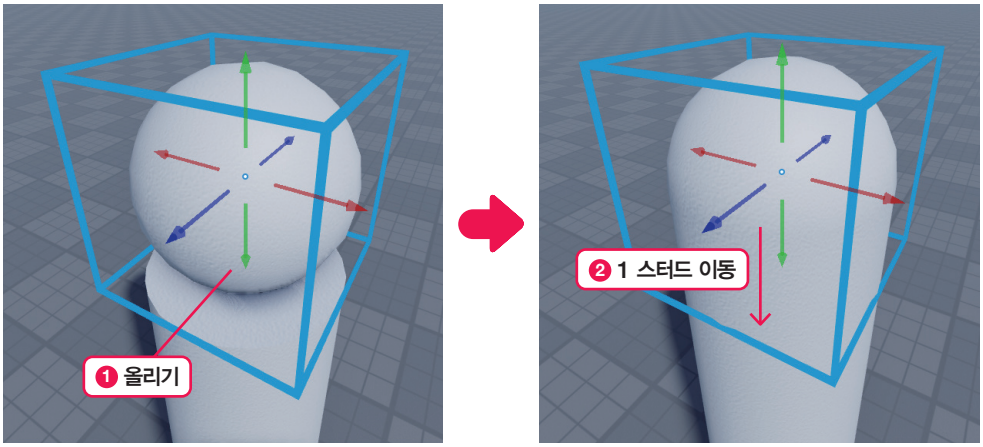


평평한 지형에서 만든 파트의 Orientation 값

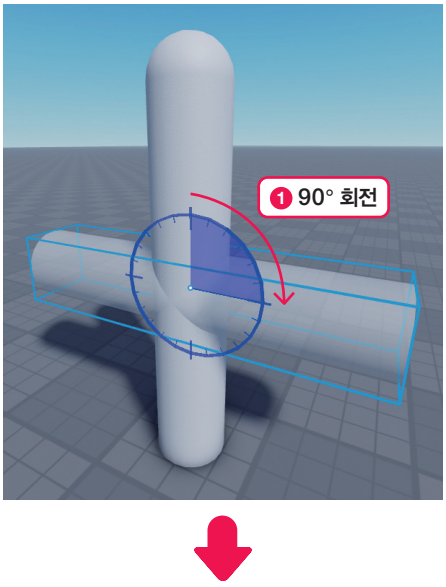
04 선인장의 윗부분을 둥글게 표현하기 위해 세워 둔 원통 파트 위에 구 파트를 겹쳐 보겠습니다. [홈] – [파트]에서 [구] 파트를 생성하고, [속성] 창의 [Transform] – [Size]를 '2, 2, 2'로 설정합니다.

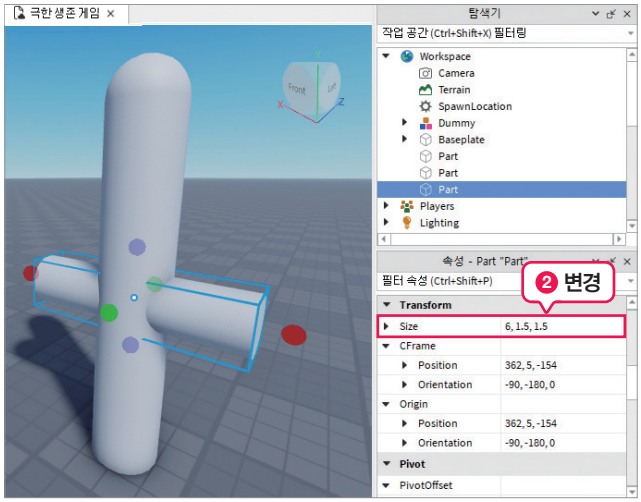


05 [홈] 탭의 [이동] 툴을 이용해 원통 파트 위에 구 파트를 올립니다. 구 파트의 위치를 1 스터드 아래로 이동시키면, 두 파트가 겹쳐져 선인장의 윗부분이 둥글게 마무리됩니다.



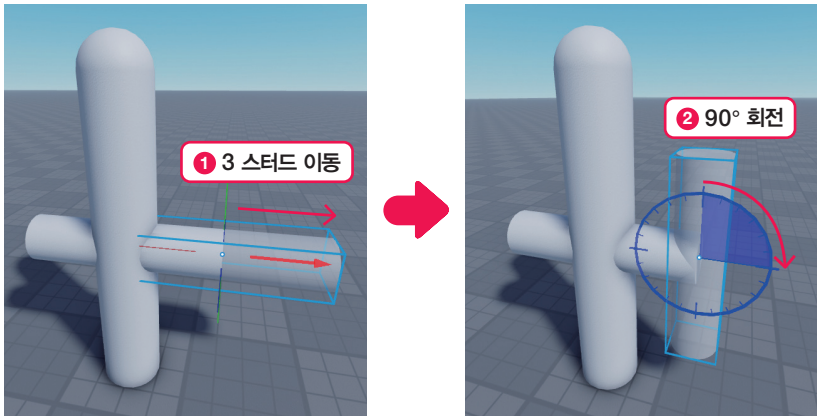
06 선인장의 팔 모양을 만들어 추가해 보겠습니다. 수직으로 세워져 있는 원통 파트를 복제하고, [회전] 툴로 90° 회전시켜 가로로 겹칩니다. 선인장의 팔을 기둥보다 얇게 표현하기 위해 [속성] 창의 [Transform] – [Size]를 '6, 1.5, 1.5'로 변경합니다.



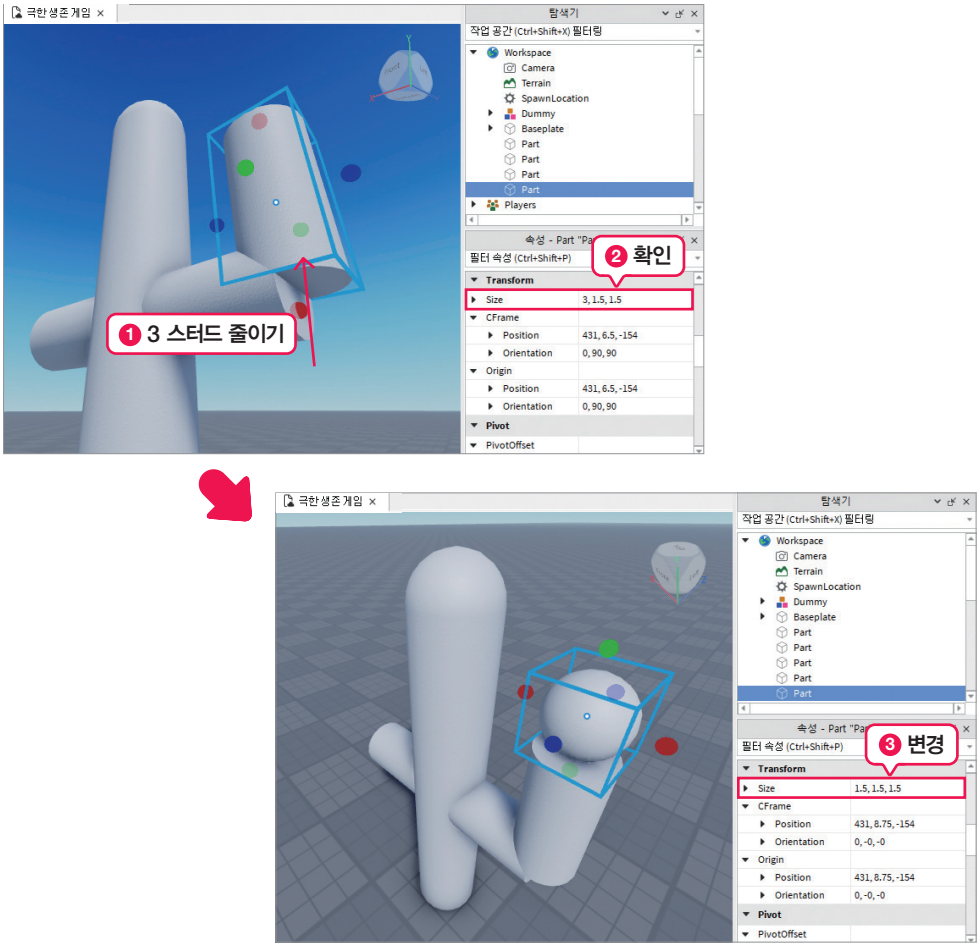


NOTE 파트를 선택하고 키보드에서 Ctrl + D를 누르면 빠르게 복제할 수 있습니다.

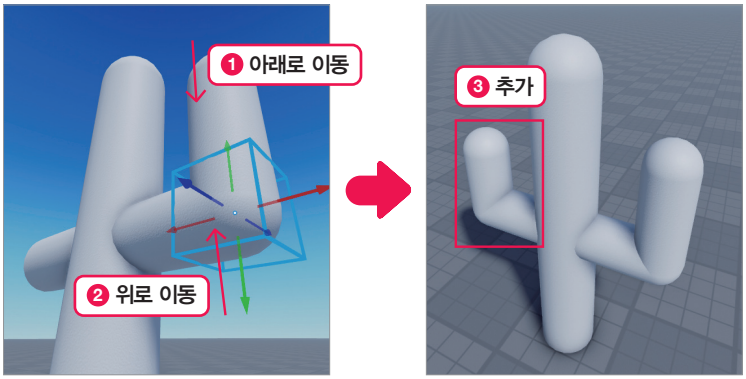
07 가로로 배치한 원통 파트를 복제하고, 복제한 파트의 중심점을 3 스터드 옆으로 옮겨 90° 회전시킵니다.



08 원통 파트의 밑 부분을 3 스터드만큼 줄이고, [Size]가 '1.5, 1.5, 1.5'인 구 파트를 생성해 양 끝에 하나씩 추가합니다.



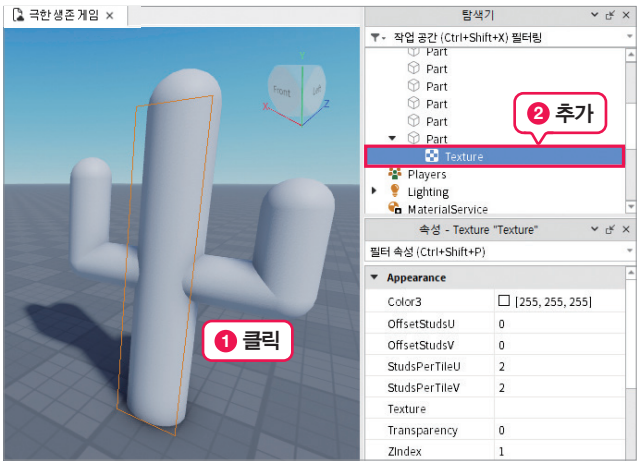
09 추가한 구 파트를 위, 아래로 0.75 스터드씩 이동시켜 원통 파트와 자연스럽게 겹쳐 주고, 동일한 방법으로 반대쪽에도 팔 모양을 추가합니다. 오른쪽보다는 파트의 길이를 좀 더 짧게 설정해 선인장의 모양을 잡아 줍니다.



STEP 2 선인장에 질감 추가하기

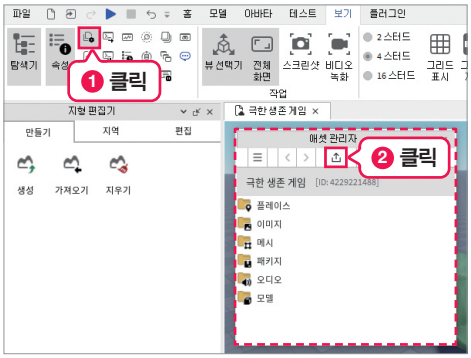
선인장 파트의 색상과 재질만 변경해도 실제 선인장과 비슷한 느낌을 낼 수 있지만, 질감 파일을 활용하면 울퉁불퉁한 선인장의 표면을 더 잘 표현할 수 있습니다.

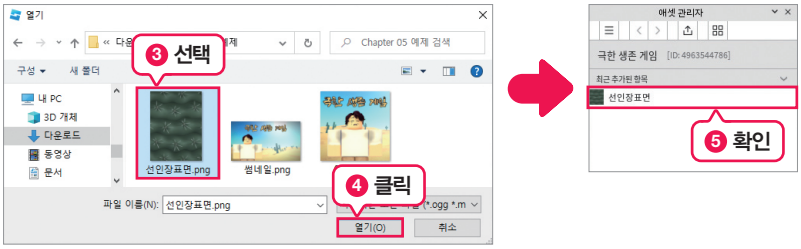
01 선인장의 기둥 파트를 클릭한 다음, [탐색기] 창에서 선택한 [Part]의 하위 항목으로 [Texture]를 추가합니다. 뷰포트에서 텍스처를 추가하면 나타나는 주황색 사각형을 클릭해 텍스처의 위치를 잡아 줍니다.



NOTE 뷰포트에서 텍스처를 추가하고 싶은 부분의 면을 정확하게 클릭하지 않으면 의도하지 않은 다른 면에 텍스처가 적용될 수 있으니 주의합니다.

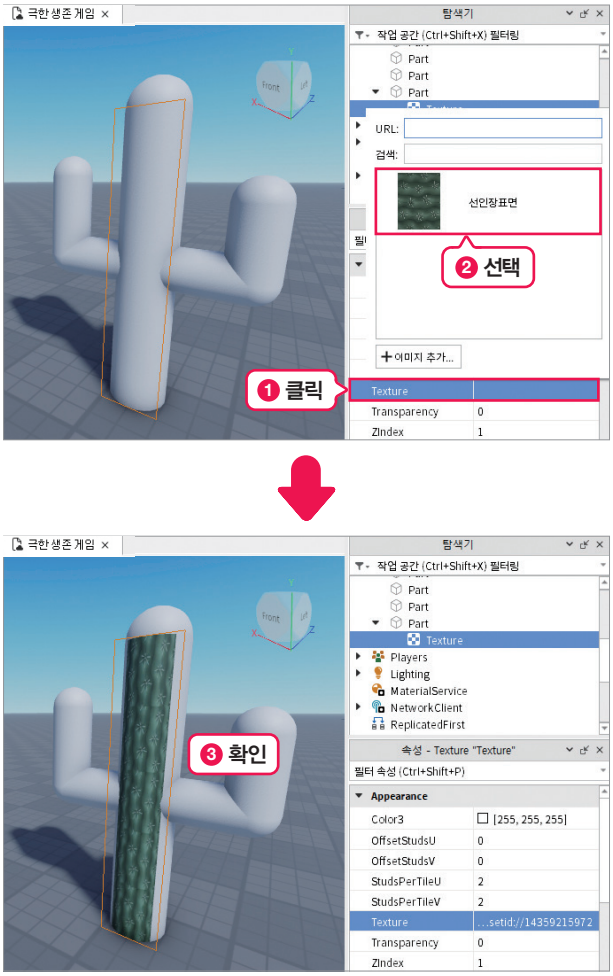
02 [보기] 탭의 [표시] 도구에서 애셋 관리자 아이콘()을 클릭해 텍스처에 추가할 이미지를 가져오겠습니다. [애셋 관리자] 창에서 일괄 가져오기 아이콘()을 클릭하고, 예제 소스의 [부록] 폴더에서 '선인장표면.png' 파일을 가져옵니다. [애셋 관리자] 창에서 [최근 추가된 항목]으로 '선인장표면' 이미지가 추가된 것을 확인할 수 있습니다.



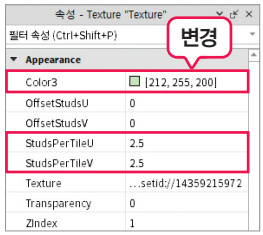


NOTE 애셋 관리자는 [파일] – [Roblox에 저장]으로 로블록스에 저장되어 있어야 사용할 수 있습니다.

03 [탐색기] 창에서 선인장 기둥 파트에 추가했던 [Texture]를 클릭합니다. [속성] 창의 [Appearance] – [Texture]를 클릭하면 나타나는 텍스처 창에서 앞에서 가져온 '선인장표면' 이미지를 선택해 추가합니다.

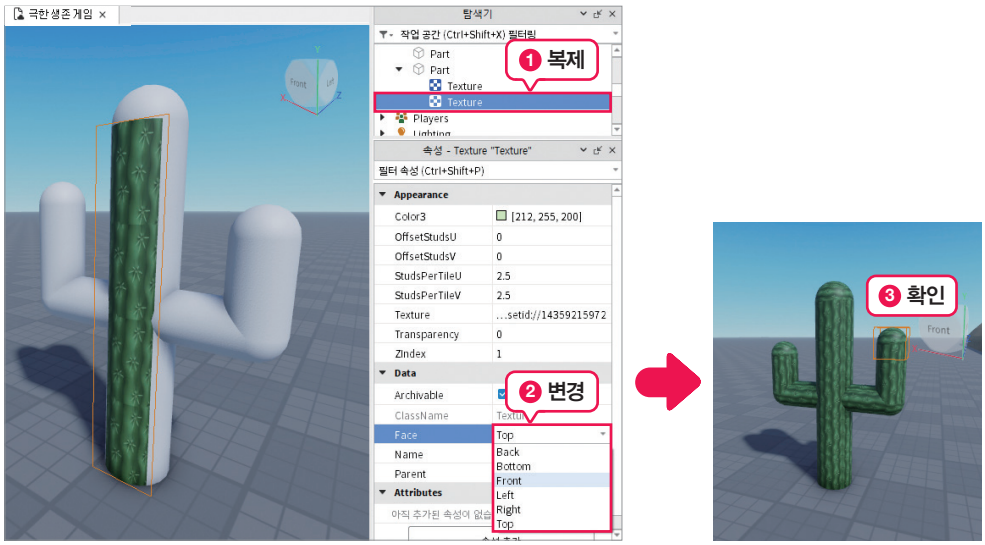


04 자연스러운 질감을 표현하기 위해 [속성] 창에서 [Appearance] – [Color3]를 '212, 255, 200'으로 변경하고, [StudsPerTileU]와 [StudsPerTileV] 값을 '2.5'로 바꿔 질감의 크기를 설정합니다.



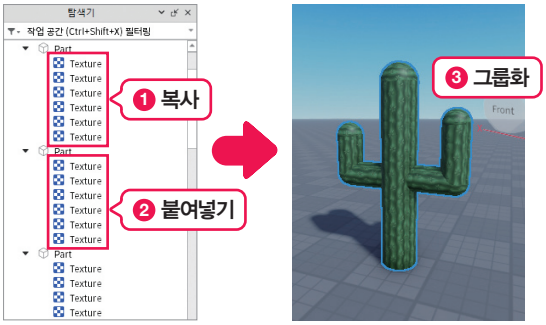
NOTE 3차원 공간에서는 X, Y, Z축으로 공간을 표현하고, 2차원 공간에서는 U(수평)와 V(수직)축으로 공간을 나타냅니다. StudsPerTileU와 StudsPerTileV는 텍스처가 적용된 타일의 크기를 2차원의 스터드 단위로 설정하는 속성입니다. StudsPerTileU로는 텍스처 타일의 수평 크기, StudsPerTileV로는 텍스처 타일의 수직 크기를 조절할 수 있습니다.

05 [탐색기] 창에서 [Texture]를 복제해 [속성] 창의 'Front'로 설정되어 있는 [Data] – [Face]의 속성을 바꿔 줍니다. 텍스처를 하나씩 복제하면서 선인장 기둥 파트의 모든 면에 질감을 추가합니다.

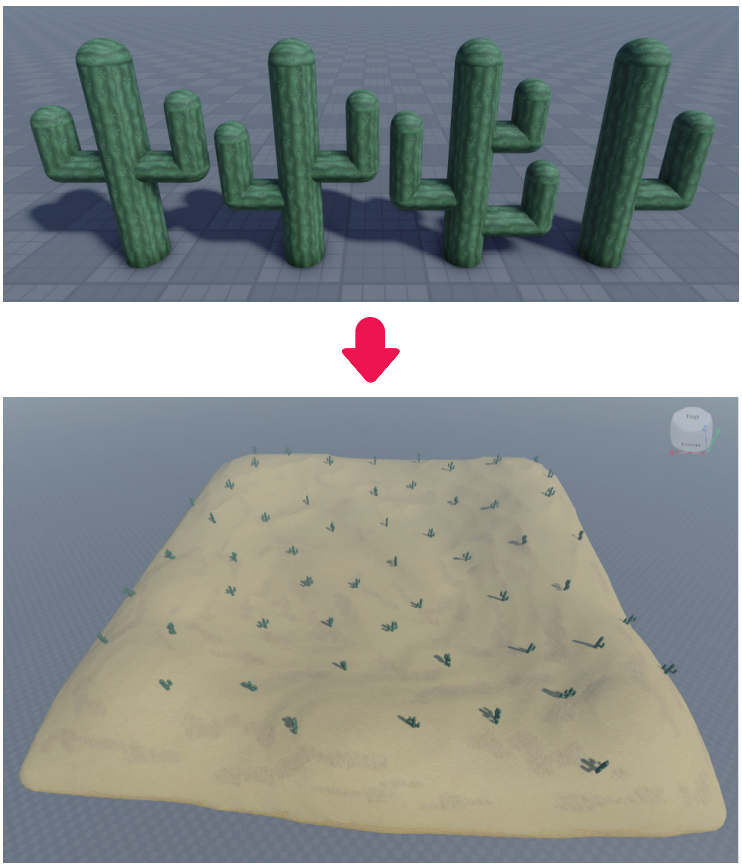


NOTE 선인장 기둥 파트의 모든 면에 텍스처를 추가하면 'Back, Bottom, Front, Left, Right, Top', 총 6개의 텍스처를 복제하게 됩니다.

06 [탐색기] 창에서 6개의 텍스처를 모두 선택해 복사하고, 나머지 팔 모양 파트에도 붙여 넣습니다. 선인장 모델을 쉽게 구분할 수 있도록 뷰포트에서 질감이 추가된 선인장 파트를 모두 선택해 Ctrl + G 키를 눌러 [모델로 그룹화]하고, 이름을 'Cactus'로 변경합니다.

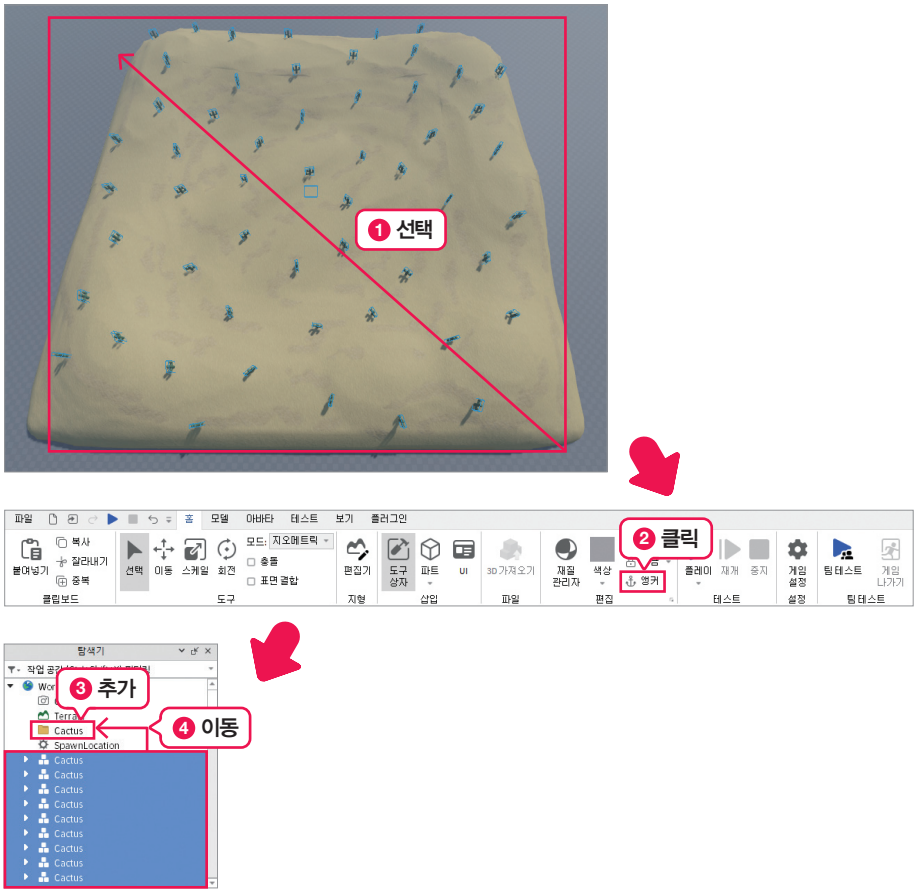


07 팔의 개수와 크기를 변경해 여러 모양의 선인장 모델을 만들면 사막 맵을 더 풍성하게 꾸밀 수 있습니다. 선인장 모델들을 복제해 사막 맵의 모래 언덕 위에 띄엄띄엄 배치합니다. [홈] 탭의 [이동], [회전] 툴을 이용해 다양한 각도와 간격으로 꾸며 줍니다.



NOTE 예제 소스의 [부록] 폴더에 있는 '선인장적용본.rbxl' 파일에서 다양한 선인장 모델로 꾸며진 사막 맵을 참고할 수 있습니다.

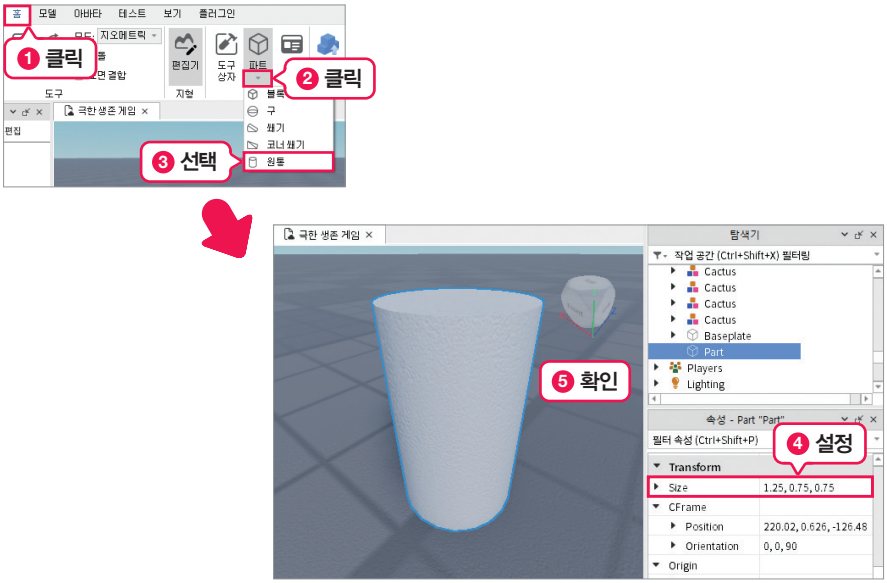
08 뷰포트에서 모든 선인장 모델을 드래그해 선택하고, [홈] 탭의 [앵커]를 클릭해 고정합니다. 마지막으로 [탐색기] 창의 [Workspace]에 [Folder]를 추가해 'Cactus'로 이름을 변경합니다. 모든 선인장 모델을 선택해 'Cactus' 폴더로 옮겨 마무리합니다.



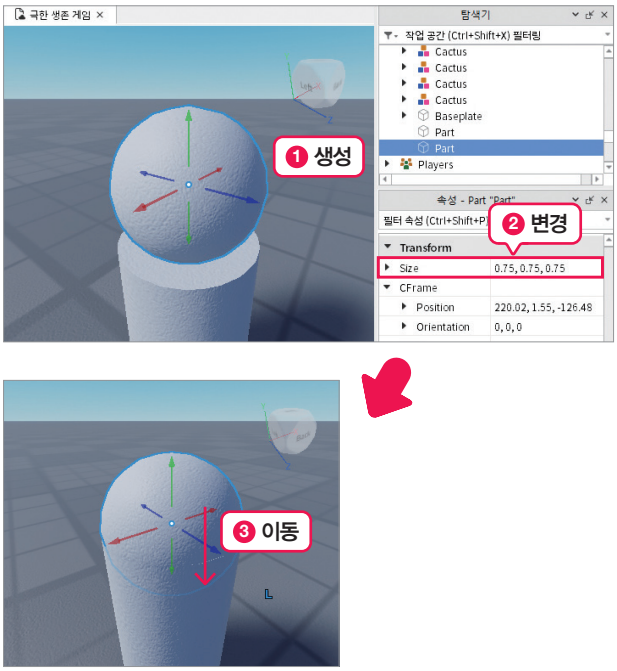
STEP 3 생수병 모델 만들기

플레이어는 뜨거운 사막에서 체력을 회복하기 위해 생수를 찾아 다니게 됩니다. 플레이어의 생명을 연장시켜 줄 회복 아이템인 생수병 모델을 만들어 보겠습니다.

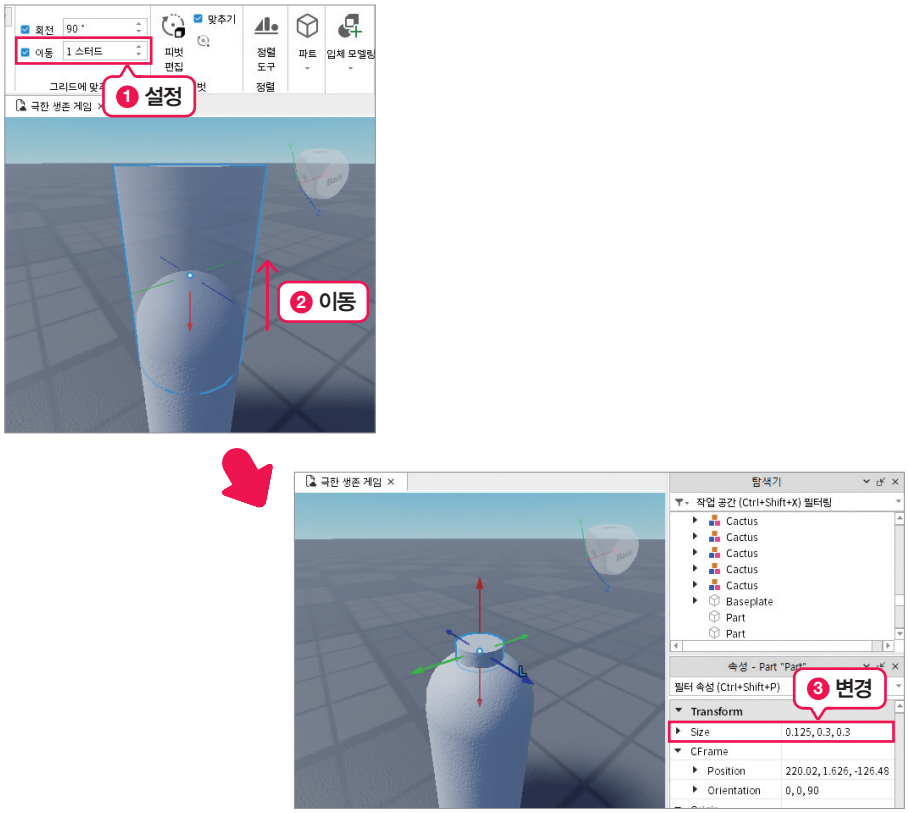
01 [홈] – [파트]에서 [원통] 파트 하나를 생성해 수직으로 세우고, [속성] 창의 [Transform] – [Size]를 '1.25, 0.75, 0.75'로 설정합니다.



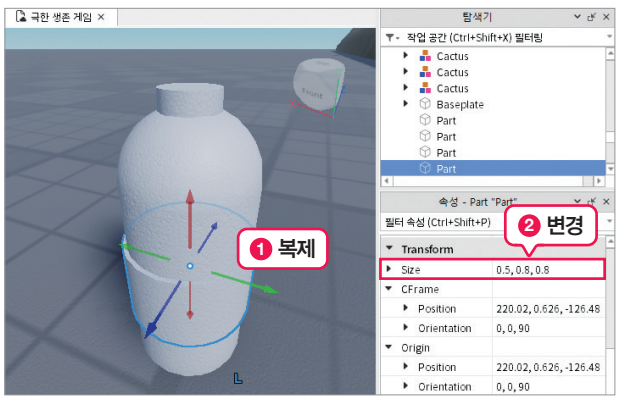
02 파트의 윗부분을 둥글게 만들기 위해 원통 파트와 같은 지름인 '0.75, 0.75, 0.75' 크기의 구 파트를 생성해 올려 놓습니다. [모델] 탭의 [이동] 단위를 조절해 아래로 이동시켜 생수병의 몸통을 완성합니다.



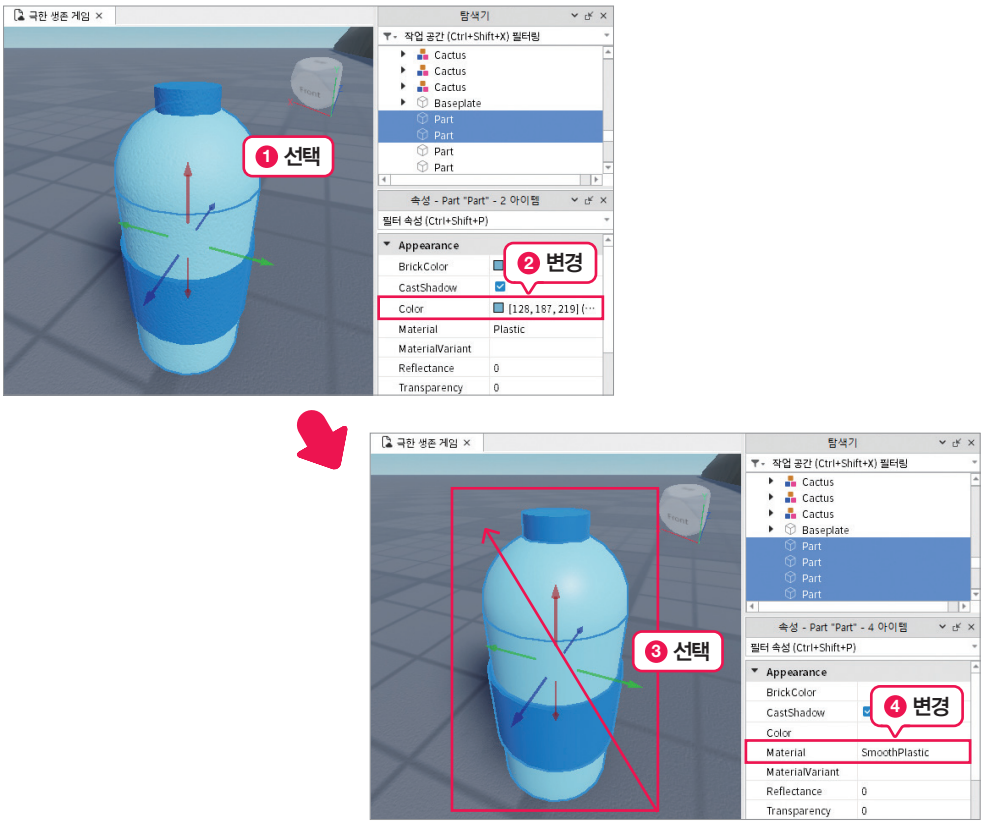
03 [모델] 탭의 [이동] 단위를 '1 스터드'로 설정한 다음, 생수병의 몸통 파트를 복제해 한 칸만 위로 옮깁니다. [속성] 창의 [Transform] – [Size]를 '0.125, 0.3, 0.3'으로 변경해 생수병의 뚜껑 모양을 만들어 줍니다.



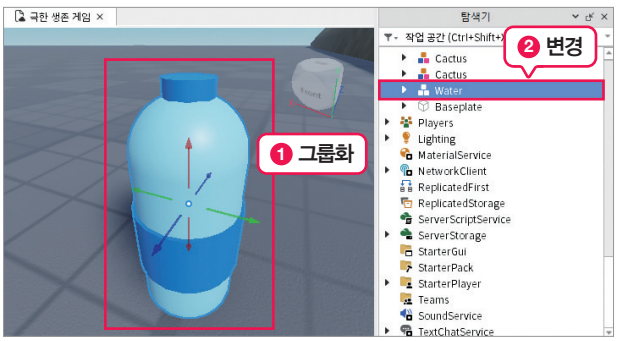
04 밋밋한 생수병 몸통에 라벨을 추가하겠습니다. 몸통 파트를 복제한 다음, [속성] 창의 [Transform] – [Size]를 '0.5, 0.8, 0.8'로 변경해 라벨의 위치를 잡아 줍니다.



05 이번에는 생수병의 색상과 재질을 바꿔 보겠습니다. 생수병의 뚜껑과 라벨 파트는 [속성] 창의 [Appearance] – [Color]를 '13, 105, 172', 몸통 파트는 '128, 187, 219'로 변경합니다. 생수병의 표면이 매끈하게 표현되도록 모든 파트를 선택하고, [Appearance] – [Material]을 'SmoothPlastic'으로 바꿔 줍니다.

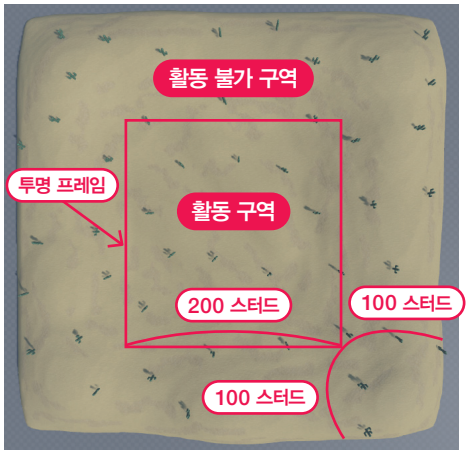


06 마지막으로 모든 생수병 파트를 선택하고 Ctrl + G 키를 눌러 [모델로 그룹화]합니다. [탐색기] 창에서 모델의 이름을 'Water'로 변경하면 완성입니다.

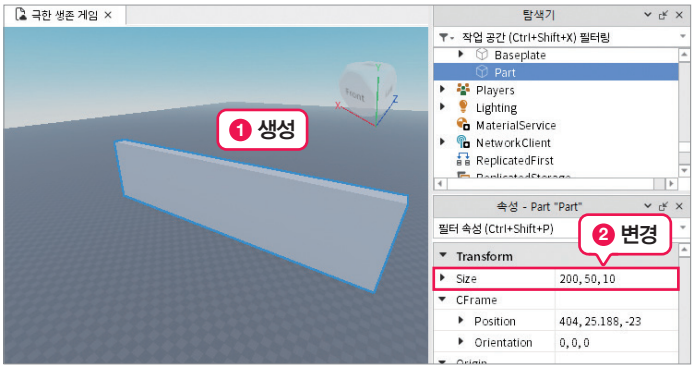


STEP 4 투명 프레임 설치하기

게임을 실행하고 플레이어가 생수를 찾아 달리다 보면 사막 맵을 벗어나 평평한 Baseplate 공간으로 나갈 수 있습니다. 우리가 기획한 뜨거운 사막의 모래 언덕 맵을 벗어나지 않도록 투명한 벽을 설치하겠습니다. 플레이어의 '활동 구역'을 전체 맵보다 작게 설정해 게임을 플레이하는 동안 플레이어의 시야에는 모래 사막만 보이도록 만듭니다.

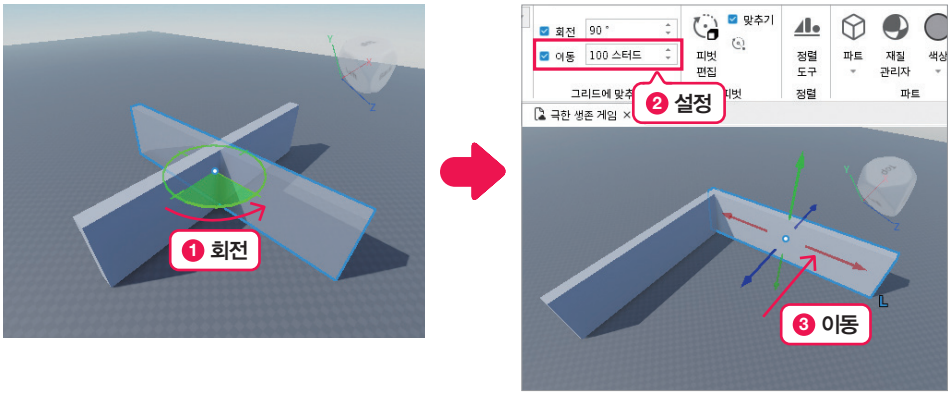


01 [홈] 탭의 [파트]에서 [블록] 파트를 생성하고, [속성] 창의 [Transform] – [Size]를 '200, 50, 10'으로 설정해 거대한 벽을 만듭니다.

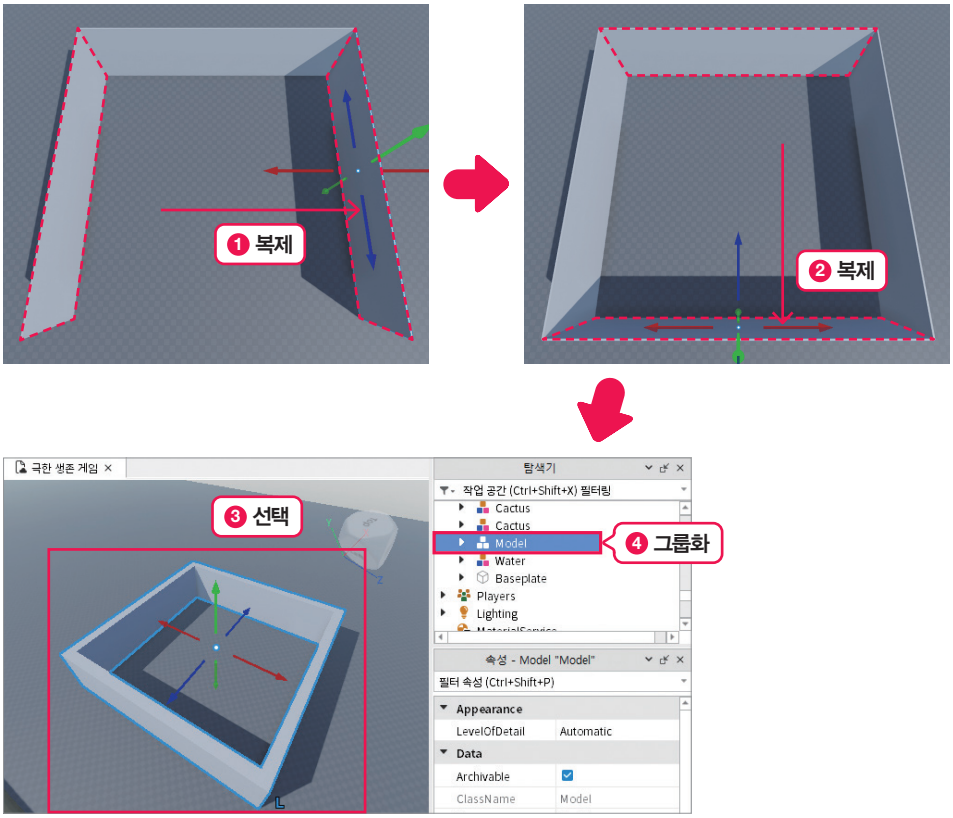


NOTE 투명한 벽의 높이는 플레이어가 게임을 플레이할 때 점프해 넘어가지 못할 정도의 높이로 설정하는 것이 좋습니다.

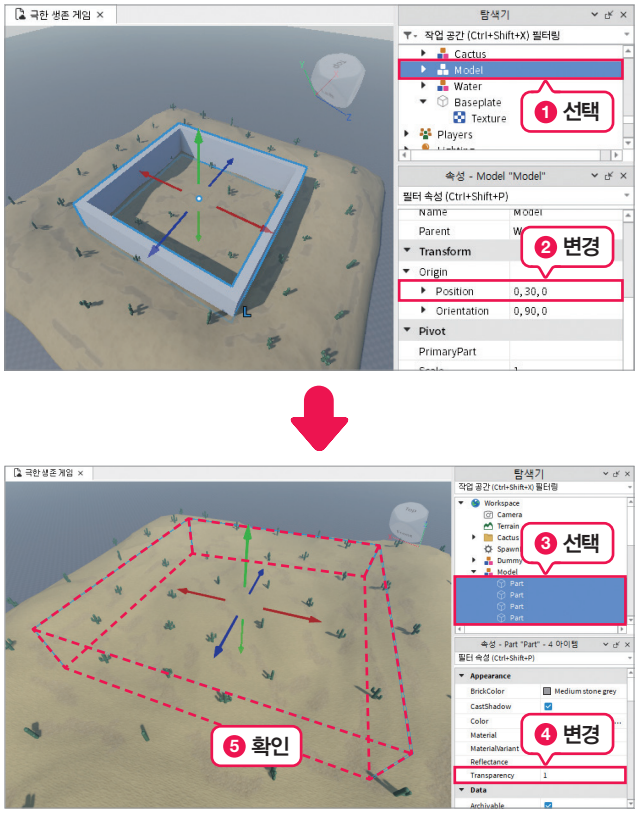
02 벽 파트를 복제하고 [회전] 툴을 이용해 90° 회전합니다. [모델] 탭의 [이동] 단위를 '100 스터드'로 설정하고, 두 파트가 ㄱ자 모양이 되도록 이동시킵니다.



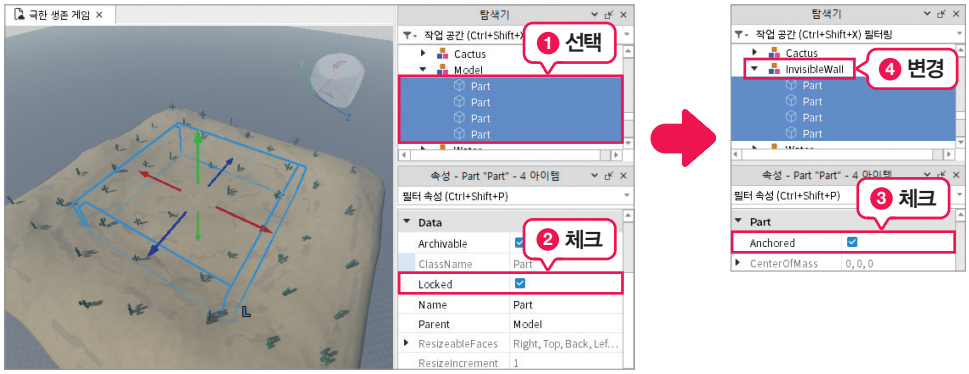
03 벽 파트를 차례로 복제해 사각형으로 배치합니다. 4개의 벽 파트를 선택하고, 한 번에 이동할 수 있도록 Ctrl + G 키를 눌러 [모델로 그룹화]합니다.



04 벽 모델을 선택하고 [속성] 창의 [Transform] – [Origin] – [Position]을 '0, 30, 0'으로 변경해 사막 맵의 중앙으로 옮깁니다. 벽 모델의 하위에 있는 모든 파트를 선택하고, [속성] 창의 [Appearance] – [Transparency]를 '1'로 설정해 투명하게 만들어 줍니다.



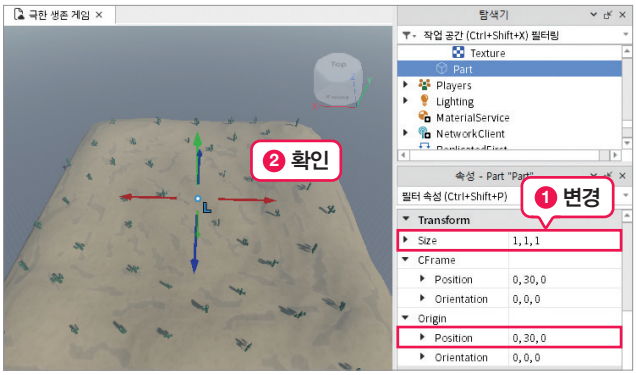
05 벽 모델이 투명해졌기 때문에 다른 작업 중 실수로 이동하지 않도록 벽 모델의 모든 파트를 선택해 [속성] 창에서 [Data] – [Locked]에 체크합니다. [Part] – [Anchored]에 체크해 고정하고, [탐색기] 창에서 벽 모델의 이름을 'InvisibleWall'로 변경해 마무리합니다.



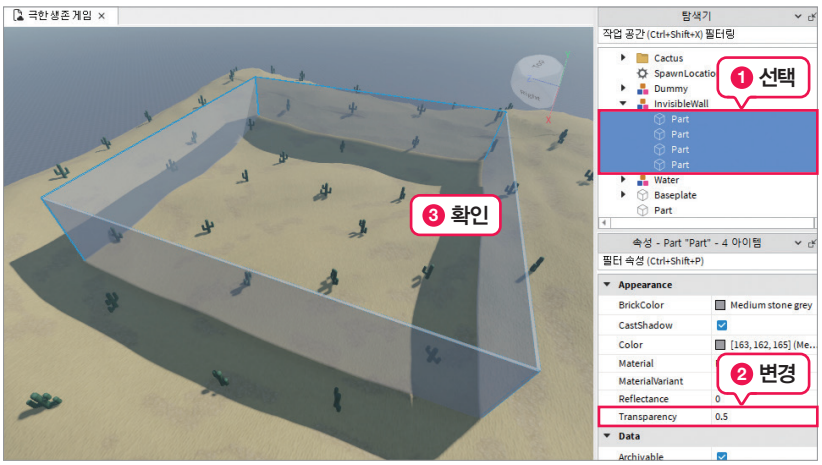
STEP 5 생수 아이템 위치 지정하기

게임을 실행했을 때 사막 맵에서 무작위로 나타나는 생수 아이템의 위치를 정해 보겠습니다. 플레이어가 사막을 뛰어다니면서 생수병을 찾는 모습을 상상하며 적절한 지점마다 등장하도록 배치하는 것이 좋습니다.

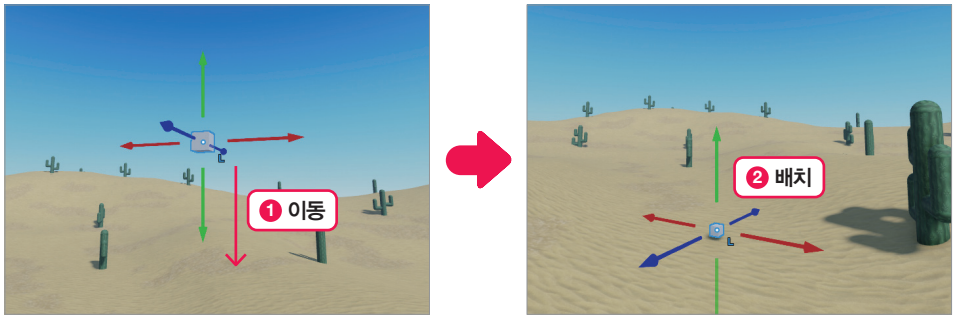
01 [홈] 탭의 [파트]에서 [블록] 파트를 생성하고, [속성] 창의 [Transform] – [Size]를 '1, 1, 1'로 설정해 작은 정사각형으로 만듭니다. [속성] 창의 [Transform] – [Origin] – [Position]을 '0, 30, 0'으로 변경해 사막 맵의 중앙으로 옮깁니다.



02 생수병들이 플레이어의 활동 구역인 투명한 벽 모델의 안쪽에서 나타나야 합니다. [탐색기] 창에서 투명한 벽 모델(InvisibleWall)의 하위에 있는 파트들을 선택하고, [속성] 창의 [Appearance] – [Transparency]를 '0.5'로 변경해 활동 구역의 범위를 확인합니다.

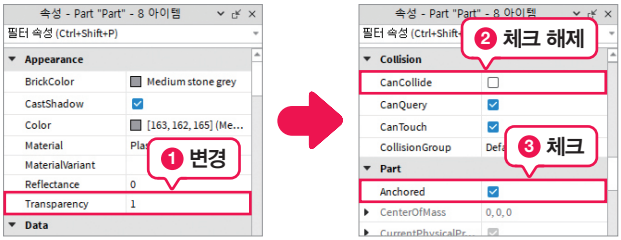


03 앞에서 만든 정사각형 파트가 배치된 지점에서 생수병 아이템이 생성, 고정되므로 파트가 공중에 떠 있지 않은지 확인해야 합니다. [홈] 탭의 [이동] 툴을 이용해 파트를 바닥과 1~2 스터드의 간격으로 이동시킵니다.



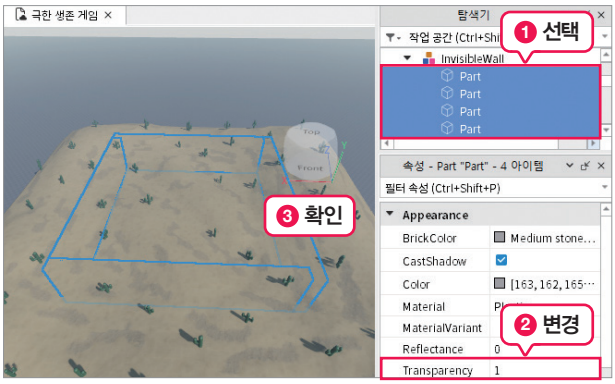
04 정사각형 파트를 총 8개로 복제해 활동 구역 안쪽에 적절하게 배치합니다. [Workspace]에 [Folder]를 생성하고 'WaterSpawner'로 이름을 변경한 다음, 모든 정사각형 파트들을 옮겨 정리합니다.

05 게임을 실행하면 보이지 않던 생수병들이 무작위로 나타나야 합니다. [탐색기] 창에서 [WaterSpawner] 폴더에 있는 모든 파트들을 선택하고, [속성] 창의 [Appearance] – [Transparency]를 '1'로, [Collision] – [CanCollide]에 체크 해제합니다. [Part] – [Anchored]에 체크해 마무리합니다.

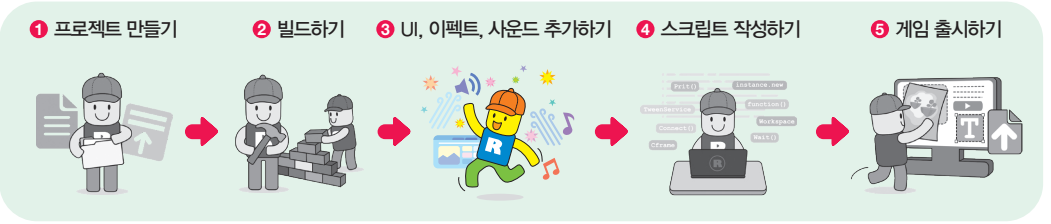


NOTE 게임을 빌드하는 마지막 단계에서는 맵을 꾸미기 위해 생성했던 모든 파트와 모델들이 잘 고정(Anchored)되어 있는지 다시 한 번 확인해 보는 것이 좋습니다.

06 마지막으로 플레이어의 활동 구역을 확인했던 벽 파트들을 다시 투명하게 바꿉니다. [탐색기] 창의 [InvisibleWall] 폴더에 있는 모든 파트들을 선택해 [속성] 창의 [Appearance] – [Transparency]를 '1'로 바꾸면 끝입니다.



A-4 UI, 이펙트, 사운드 추가하기



이제 맵에 UI와 효과 이펙트, 배경 음악과 사운드를 추가해 더 멋지게 완성할 차례입니다. 게임의 규칙을 알려주는 UI와 사막 환경에 어울리는 조명 및 아이템, 게임을 더 실감 나게 만들어 주는 효과 사운드를 하나씩 추가해 보겠습니다.

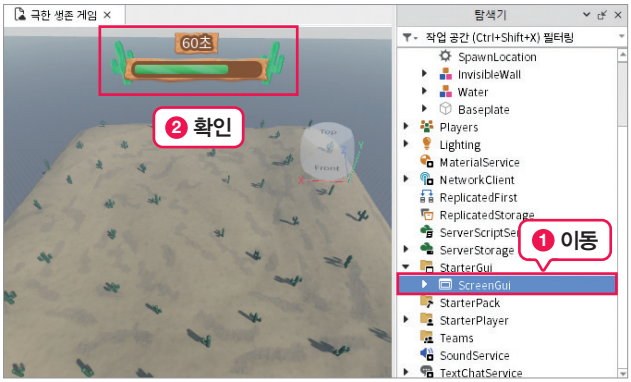
게임 규칙 UI 추가하기

극한 생존 게임에는 게임의 제한 시간이나 생수를 마시고 회복되는 플레이어의 체력, 게임의 성공 여부와 같은 규칙이 있습니다. 플레이어가 이 규칙들을 확인하면서 게임을 실행해야 하므로 모두 게임 화면에 표시되어야 하는 UI에 해당합니다.

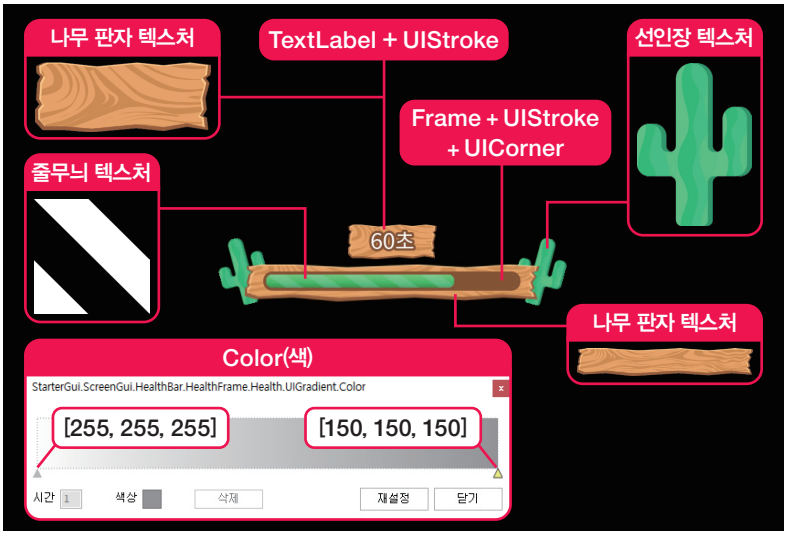
STEP 1 제한 시간 & 체력 회복 UI

플레이어가 남은 시간 동안 얼만큼의 체력을 회복하면서 살아남아야 하는지 화면에 표시해 보겠습니다. 게임을 실행하는 내내 한눈에 확인할 수 있도록 화면의 상단, 중간 지점에 위치를 잡겠습니다.

01 예제 소스의 [부록] 폴더에서 '극한생존게임UI.rbxm' 파일을 뷰포트 화면으로 가져옵니다. [탐색기] 창의 [Workspace]에 추가된 UI인 [ScreenGUI]를 [StarterGui]로 폴더로 옮겨 확인합니다.



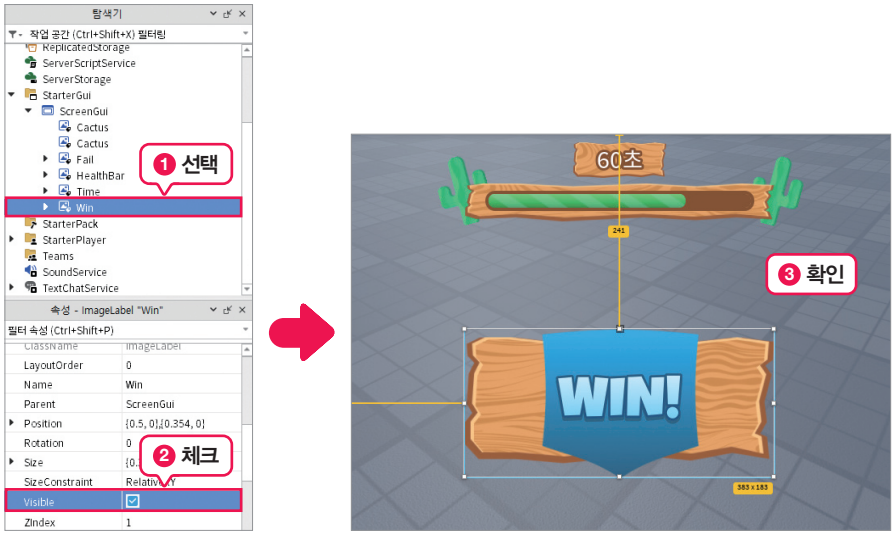
02 추가된 UI는 제한 시간 UI와 체력 회복 UI로 구성되어 있는 것을 확인할 수 있습니다. 시각적으로 게임의 규칙과 사막 맵의 분위기를 잘 표현하기 위해 여러 속성들이 포함되어 있습니다.



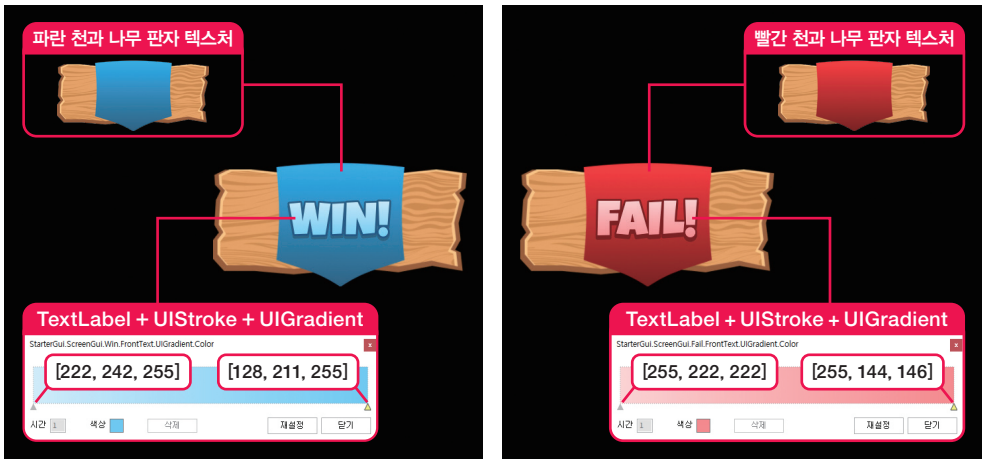
STEP 2 성공 & 실패 UI

게임이 종료됐을 때 플레이어에게 제한 시간 내에 체력을 회복해 살아남았는지 실패했는지 게임의 결과를 알리려고 합니다. 간단한 UI를 추가해 봅시다.

01 앞에서 가져온 '극한생존게임UI.rbxm' 파일에는 'Win', 'Fail' UI도 포함되어 있습니다. [탐색기] 창에서 UI를 선택하고 [속성] 창의 [Data] – [Visible]를 체크하면 뷰포트에서 UI를 확인할 수 있습니다.



02 UI를 살펴보겠습니다. 성공과 실패를 표현하기 위해 각각 파란색과 빨간색으로 구분하고, 여러 속성들이 포함되어 있는 것을 확인할 수 있습니다.

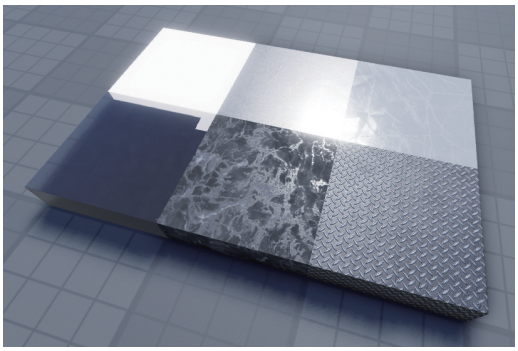


라이팅 이펙트 추가하기

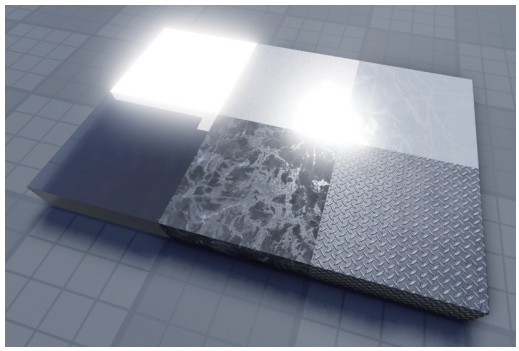
극한 생존 게임의 배경은 태양이 내리쬐는 사막이기 때문에 다양한 이펙트를 추가하면 게임을 즐기는 플레이어들이 훨씬 더 재미있게 게임에 빠져들 수 있습니다. 게임에 각종 조명 효과를 넣는 여러 가지 라이팅(Lighting) 이펙트를 추가해 보겠습니다.

STEP 1 SunRays & Bloom 이펙트

SunRays(태양 광선) 이펙트는 하늘에서 비치는 태양의 빛줄기를 표현하고, Bloom(광원) 이펙트는 빛을 받는 물체의 주위로 발산되는 빛을 표현합니다. Bloom 이펙트의 경우 빛이 반사되는 효과를 주기 때문에, 특히 오브젝트의 재질이 'Metal, Neon, Ice, Glass, Granite, DiamondPlat'인 경우에 더 잘 나타납니다. 두 이펙트를 사용해 강렬하게 비추는 쨍쨍한 사막의 햇빛을 표현해 보겠습니다.



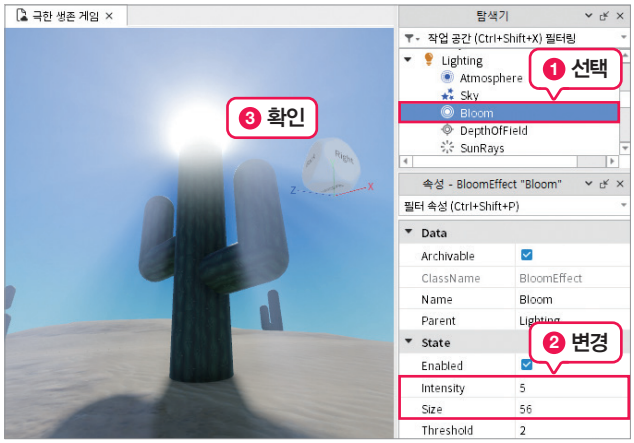
Bloom 이펙트를 적용하지 않았을 때



Bloom 이펙트를 적용했을 때

01 추가할 이펙트의 효과를 바로 확인할 수 있도록 뷰포트를 태양이 선인장 뒤편에서 살짝 가려진 시점으로 조정합니다. 그리고 [탐색기] 창의 [Lighting]에 있는 'SunRays' 이펙트를 선택해 [속성] 창의 [State] – [Intensity(강도)]를 '0.25', [Spread(확산)]를 '1'로 변경합니다. 선인장 뒤로 비치는 태양 광선이 더 강하고, 넓게 퍼지는 것을 확인할 수 있습니다.

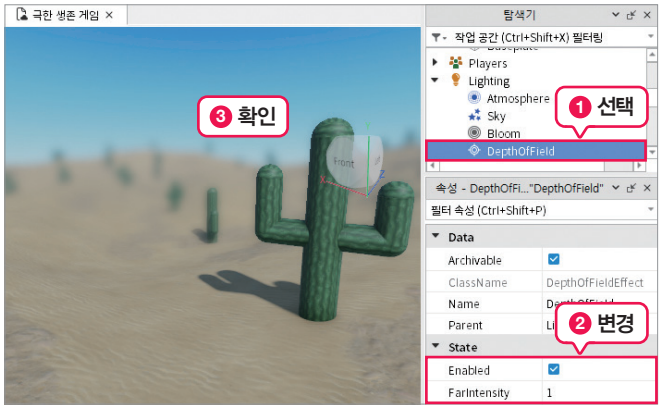
02 이번에는 Bloom 이펙트를 설정해 보겠습니다. [탐색기] 창에서 [Lighting]에 있는 'Bloom' 이펙트를 선택해 [속성] 창의 [State] – [Intensity]를 '5', [Size]를 '56'으로 바꿉니다. 광원인 태양이 더 크고, 넓게 반사되는 것을 확인할 수 있습니다.



STEP 2 DepthOfField & ColorCorrection 이펙트

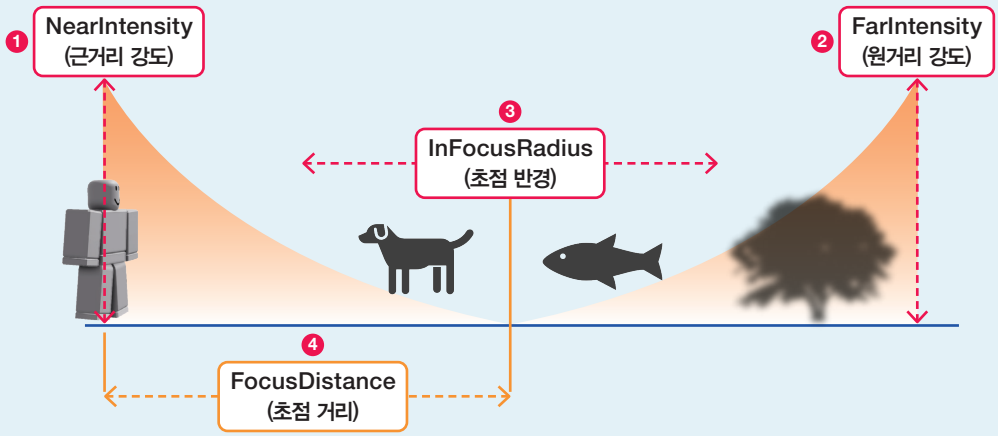
DepthOfField 이펙트로는 오브젝트에 흐림 효과를 적용해 원근감을 표현하고, ColorCorrection (색상 보정) 이펙트로는 게임 화면의 밝기와 대비, 채도, 색상 등을 조절합니다. 두 이펙트를 사용해 뜨거운 열기로 멀리 있는 선인장이 흐릿하게 보이는 한낮의 사막을 표현해 보겠습니다.

01 [탐색기] 창에서 [Lighting]에 있는 'DepthOfField' 이펙트를 선택합니다. [속성] 창의 [State] – [Enabled]에 체크해 이펙트를 활성화하고, [State] – [FarIntensity (원거리 강도)]를 '1'로 바꾸면 멀리 있는 선인장들이 흐릿하게 표현되는 것을 확인할 수 있습니다.



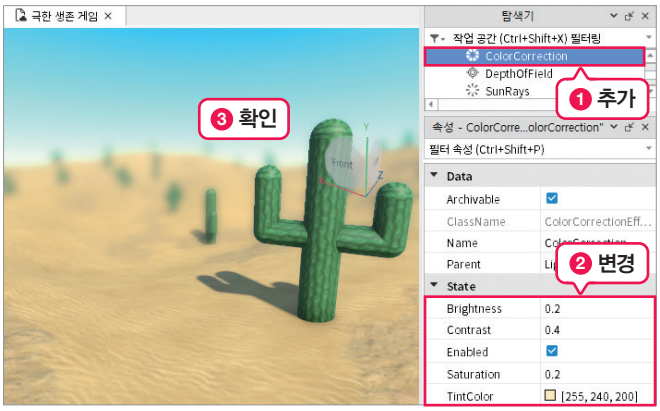
여기서 잠깐 DepthOfField 이펙트의 다른 속성

DepthOfField 이펙트에는 앞에서 설정한 [FarIntensity] 외에도 여러 속성들이 있습니다. 각각의 속성을 세밀하게 조정하면 맵을 좀 더 섬세하게 표현할 수 있습니다.



1. **NearIntensity(근거리 강도):** 가까이 있는 오브젝트의 흐림 효과를 조절합니다.
2. **FarIntensity(원거리 강도):** 멀리 있는 오브젝트의 흐림 효과를 조절합니다.
3. **InFocusRadius(초점 반경):** 초점의 반지름을 나타내는 숫자가 클수록 초점의 범위, 즉 선명하게 보이는 부분이 커집니다. 반대로 숫자가 작을수록 선명하게 보이는 범위가 작아지므로 이번에는 흐리게 보이는 부분이 커집니다.
4. **FocusDistance(초점 거리):** 선명하게 보이는 초점의 거리를 가깝게 둘지, 멀리 둘지를 조절합니다.

02 [탐색기] 창의 [Lighting]에 [ColorCorrectionEffect]를 추가하고, [속성] 창에서 [State] – [Brightness(밝기)]를 '0.2', [Contrast(대조)]를 '0.4', [Saturation(채도)]을 '0.2', [TintColor(색조)]를 '255, 240, 200'으로 설정해 게임의 화면을 더 밝고 쨍하게 바꿉니다.



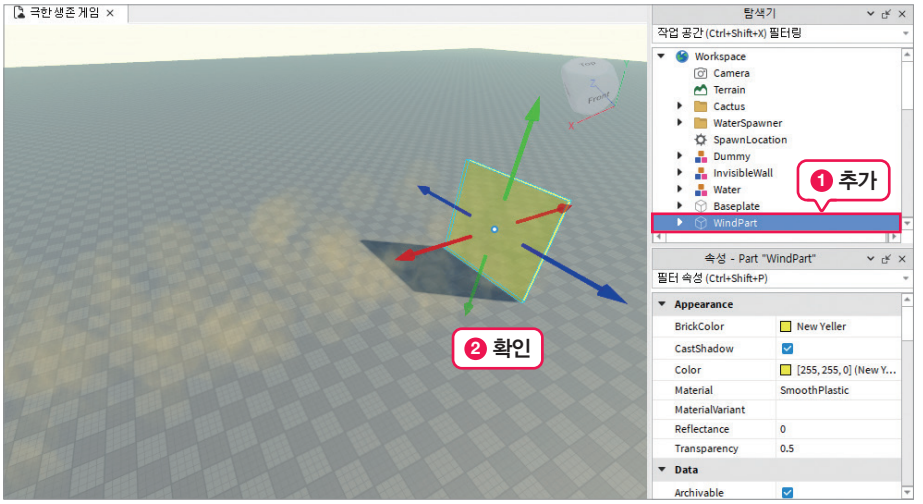
파티클 이펙트 추가하기

사막의 모래 언덕에는 휘몰아치는 모래바람도 있습니다. 사막 맵에 있는 플레이어가 흐릿한 모래 언덕 너머로 반짝이는 생수병을 발견할 수 있도록 파티클 이펙트를 더해 보겠습니다.

STEP 1 모래바람 이펙트

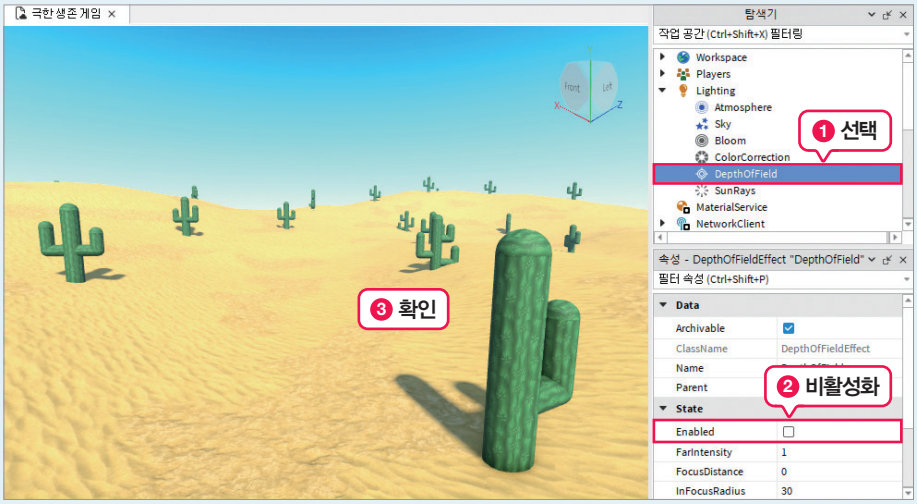
사막을 뒤덮는 모래바람을 이펙트로 표현해 보겠습니다. ParticleEmitter 이펙트가 포함된 파트를 설치하고 플레이어에게는 보이지 않게 설정하는 방법입니다.

01 예제 소스의 [부록] 폴더에서 '모래바람이펙트.rbxm' 파일을 가져오면 반투명한 노란색 파트와 모래바람 이펙트를 확인할 수 있습니다. 사막 맵 바깥의 Basaplate로 파트를 가져오면 이펙트의 효과가 더 잘 보입니다.

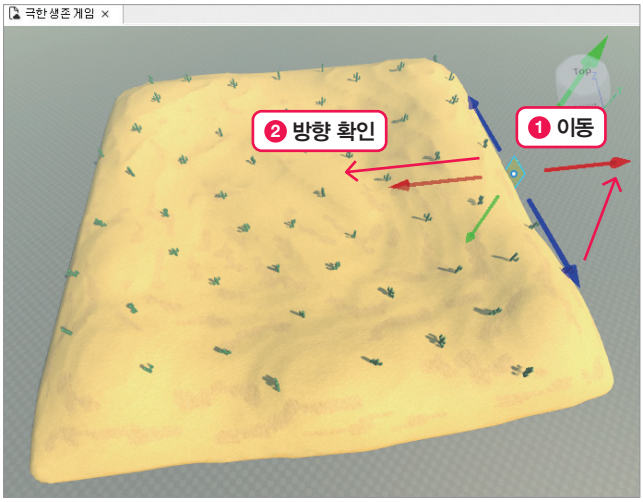


여기서 잠깐 **라이팅 이펙트 끄고 작업하기**

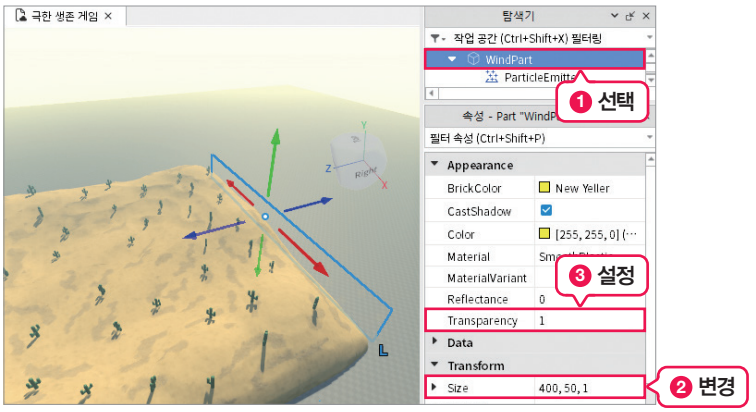
시각 효과인 라이팅 이펙트를 설정하면 다른 작업을 할 때 조금 불편할 수 있습니다. 앞에서 설정한 DepthOfField 이펙트의 효과로 뷰포트 화면이 흐릿하게 보여 모래바람 이펙트를 확인하기 어렵기 때문입니다. 이럴 때는 [탐색기] 창의 [Lighting]에 있는 'DepthOfField' 이펙트를 선택하고, [속성] 창의 [State] – [Enabled]를 잠깐 비활성화하면 작업이 좀 더 편리합니다.



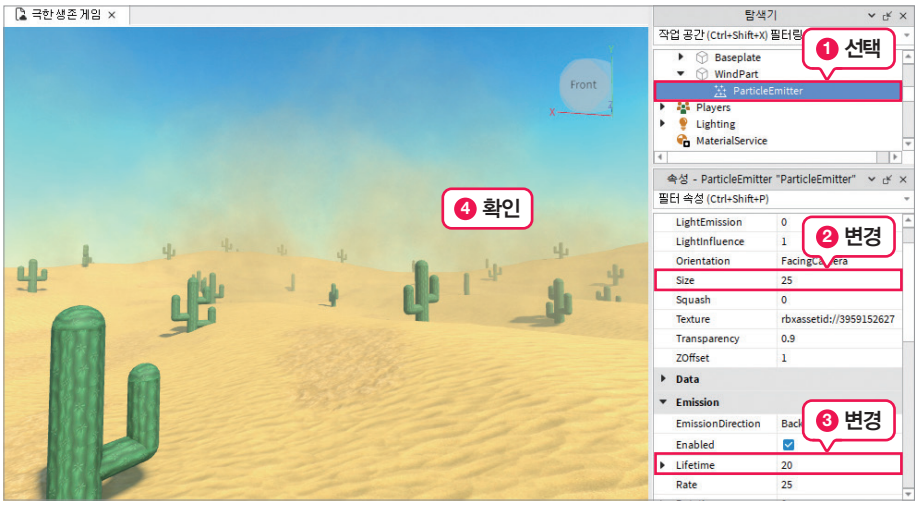
02 사막 맵에 모래바람 이펙트가 적용될 수 있도록 이펙트 파트를 사막 맵 옆으로 이동시킵니다. 모래바람의 방향이 사막 맵 쪽을 향할 수 있도록 파트의 방향에 주의합니다.



03 사막 맵 전체에 모래바람이 뒤덮일 수 있도록 파트의 크기와 이펙트의 범위를 넓히겠습니다. [탐색기] 창에서 [Workspace] – [WindPart]를 선택해 [속성] 창의 [Transform] – [Size]를 '400, 50, 1'로 변경하고, [Appearance] – [Transparency]를 '1'로 설정해 보이지 않게 숨겨 줍니다.

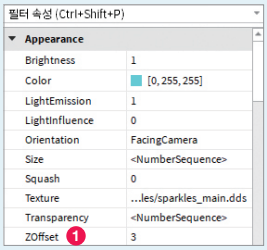


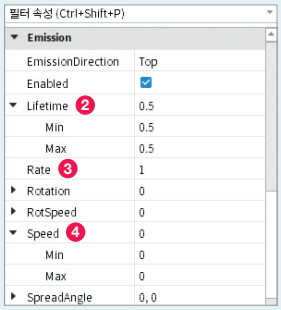
04 [WindPart]에 속해 있는 [ParticleEmitter]의 [속성] 창에서 [Appearance] – [Size]를 '25', [Emission] – [LifeTime]을 '20'으로 설정해 이펙트의 크기를 정하면 끝입니다.



여기서 잠깐 ParticleEmitter 이펙트의 다른 속성

ParticleEmitter 이펙트에는 앞에서 설정한 [LifeTime] 외에도 여러 속성들이 있습니다. 각각의 속성을 세밀하게 조정하면 맵을 좀 더 섬세하게 표현할 수 있습니다.



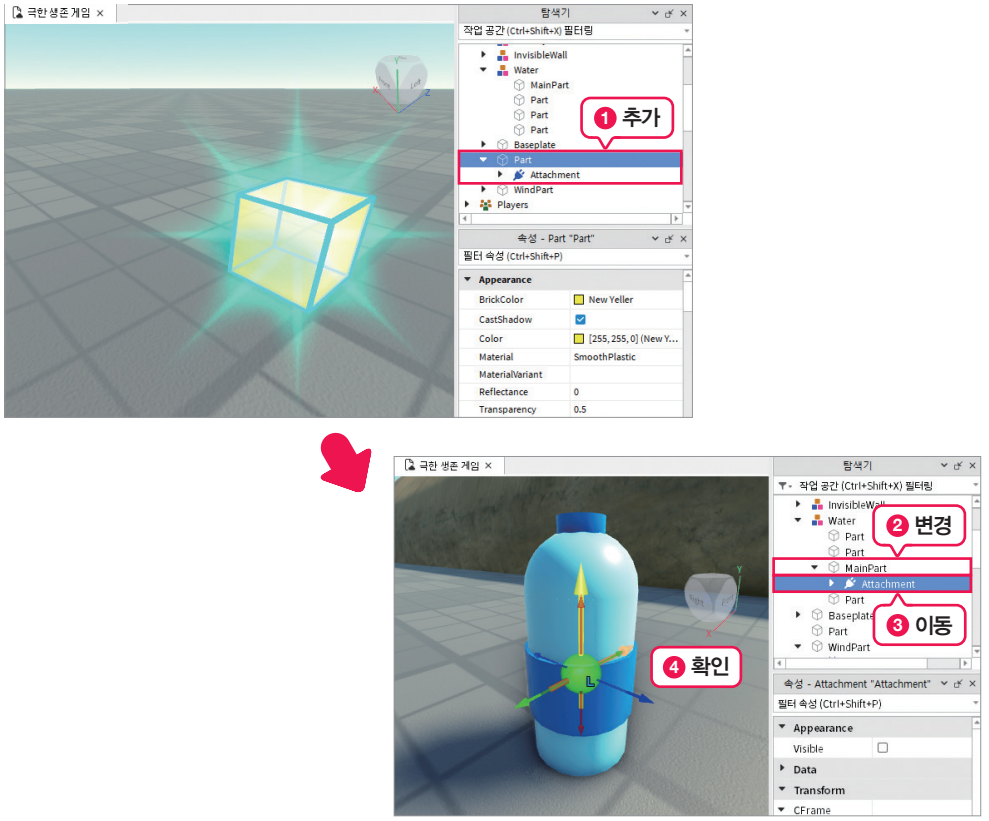


1. **ZOffset:** 보고 있는 화면의 시점에서 파티클의 앞뒤(Z) 방향의 위치를 조절합니다. 파티클의 ZOffset 값이 작을수록 파티클이 다른 오브젝트보다 뒤에 있게 되고, ZOffset 값이 클수록 다른 오브젝트보다 앞에 있도록 설정됩니다.
2. **LifeTime:** 파티클이 시작하는 곳에서부터 끝나는 지점까지의 수명을 조절합니다. [속성] 창에서 [Emission] – [LifeTime] 왼쪽에 있는 더보기 아이콘을 클릭하면, 파티클 수명의 최솟값(Min)과 최댓값(Max)을 설정할 수 있습니다.
3. **Rate:** 1초 동안 생성되는 파티클의 양을 조절합니다. Rate 값이 1이면 1초에 한 번씩 파티클이 생성되고, 2이면 1초에 두 번씩 파티클이 생성됩니다.
4. **Speed:** [EmissionDirection]에서 설정한 방향으로 파티클이 이동하는 속도를 조절합니다. 최솟값(Min)과 최댓값(Max)을 다르게 설정하여 생성되는 파티클의 속도를 설정할 수 있습니다.

STEP 2 반짝임 이펙트

생수병 아이템이 랜덤으로 생성될 때 플레이어가 생수병이 어디에서 생성됐는지 알 수 있도록 반짝이는 이펙트를 구현해 보겠습니다. 생수병에 반짝이는 빛이 점점 커졌다가 사라지도록 만들어 봅시다.

01 예제 소스의 [부록] 폴더에서 '생수병이펙트.rbxm' 파일을 가져오면 반짝임 이펙트가 포함된 반투명한 정사각형 파트를 확인할 수 있습니다. 앞에서 만들었던 생수병 모델에 이펙트를 적용하기 위해 몸통 파트의 이름을 'MainPart'로 바꾸고, 이펙트 파트에 있는 [Attachment]를 [MainPart]로 옮깁니다.



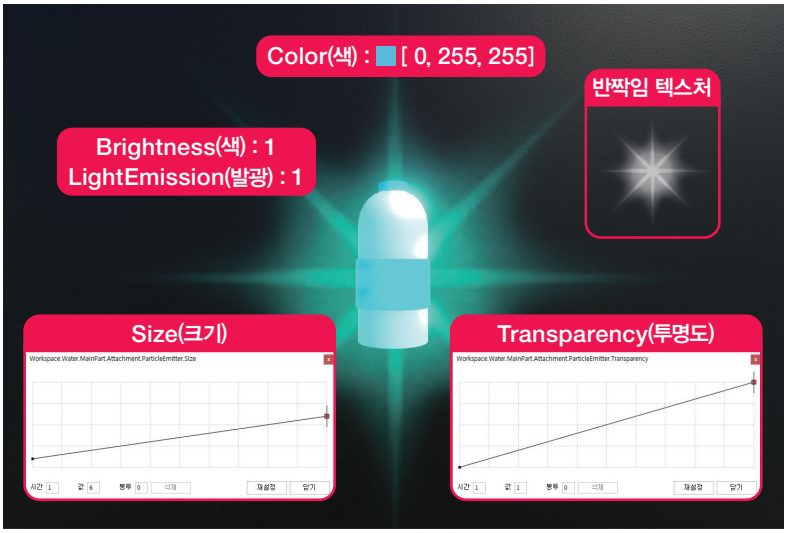
NOTE 생수병 모델은 [탐색기] 창에서 [Water] 모델을 선택하고, 키보드에서 F 키를 눌러 찾을 수 있습니다.

여기서 잠깐 이펙트에 Attachment가 필요한 이유

왜 반짝임 이펙트는 모래바람 이펙트처럼 [ParticleEmitter]를 추가하지 않고 [Attachment]를 이용할까요? 그림과 같이 고정된 Attachment가 없는 파트는 파티클이 파트 곳곳에 흩어져 생기고, Attachment가 있는 파트는 파티클이 파트의 중심점을 기준으로 생기는 것을 볼 수 있습니다.

별도의 설정을 하지 않으면 파트에 포함된 Attachment는 뷰포트에 표시되지 않습니다. [모델] 탭에서 [제약 세부 사항] 아이콘()을 클릭해 활성화하면 초록색 점으로 빛나는 Attachment의 중심점이 표시되며, [이동], [회전] 툴을 이용해 이펙트의 위치와 방향을 바꿀 수 있습니다.

02 반짝임 이펙트를 살펴보면 색상, 텍스처, Brightness(밝기), LightEmission(발광) 등의 다양한 속성들이 포함되어 있습니다.



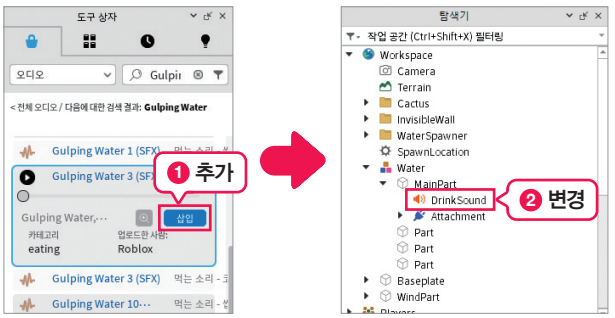
03 파티클 이펙트 추가를 완료했으므로 [탐색기] 창의 [Lighting]에 있는 'DepthOfField' 이펙트를 선택하고, [속성] 창의 [State] – [Enabled]를 활성화해 다시 라이팅 이펙트를 적용합니다.

효과 사운드 추가하기

게임의 배경이나 상황에 맞는 사운드를 추가하면 더 재미있는 게임으로 만들 수 있습니다. 사막 맵에는 어떤 효과 사운드가 필요할까요? 앞에서 사막을 뒤덮는 모래바람 이펙트와 생수병에 반짝임 이펙트를 추가했으니 그에 어울리는 바람 소리와 꿀꺽 꿀꺽 물 마시는 소리를 추가해 보겠습니다.

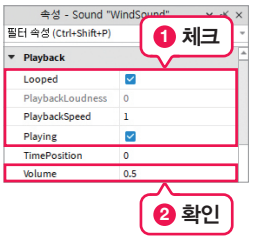
01 먼저 바람 소리부터 추가해 보겠습니다. [홈] 탭에서 [도구 상자]를 클릭해 창을 열고, [오디오] 카테고리에서 'Spiraling Wind'를 검색해 원하는 바람 소리를 찾습니다. [탐색기] 창의 [SoundService]를 클릭하고 [도구 상자]에서 원하는 오디오를 선택해 오디오를 삽입합니다. 바람 소리 오디오의 이름을 'WindSound'로 변경해 정리합니다.

02 같은 방법으로 물 마시는 소리도 추가해 보겠습니다. [도구 상자]에서 'Gulping Water'를 검색하고, 원하는 오디오를 찾아 [탐색기] 창의 [Workspace]에 있는 [Water] 모델의 'MainPart'에 추가합니다. 그리고 물 마시는 소리 오디오의 이름을 'DrinkSound'로 변경해 정리합니다.



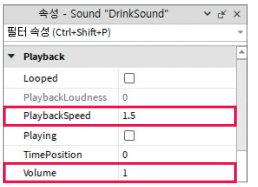
NOTE 물 마시는 소리를 바람 소리처럼 [SoundService]에 저장하지 않는 이유가 있습니다. [SoundService]에 저장된 물 마시는 소리를 제대로 재생하려면 RemoteEvent를 활용해 복잡한 코드를 작성해야 합니다. 따라서 쉬운 방법으로, 플레이어가 생수병 모델을 터치했을 때 소리가 재생될 수 있도록 [Water] 모델 안에 'DrinkSound'를 저장했습니다.

03 게임을 플레이했을 때 바람 소리가 반복 재생되도록 [탐색기] 창에서 [SoundService] – [WindSound]를 선택하고, [속성] 창에서 [Playback] – [Looped]와 [Playing]에 체크합니다. 바람 소리가 잘 들리는지 확인해 보고, 소리가 작다면 [Volume]의 값을 키워 '1' 정도로 변경해도 괜찮습니다.

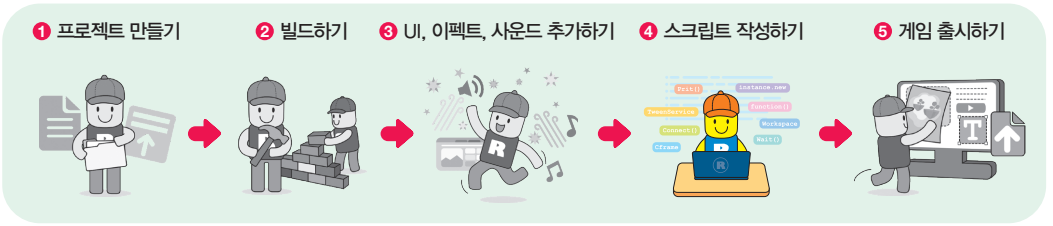


NOTE [Looped]는 사운드를 무한하게 반복 재생하는 속성이고, [Playing]은 게임의 시작과 동시에 사운드가 실행되도록 설정하는 속성입니다.

04 물을 마시는 소리는 바람 소리처럼 계속 재생되어야 하는 사운드가 아니므로 [Looped]와 [Playing] 속성을 활성화하지 않습니다. 대신 [PlaybackSpeed] 값을 '1' 이상으로 올려 사운드의 재생 배속을 높이거나, [Volume] 값을 변경해 소리의 크기를 조절할 수 있습니다.



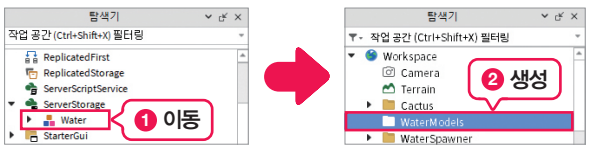
A-5 스크립트 작성하기



지금까지 만든 맵과 이펙트를 적용하기 위해서는 각각에 맞는 스크립트를 작성해야 합니다. 먼저 생수 아이템이 사막에서 무작위로 나타났다 사라지는 기능을 적용하고, 플레이어의 체력 감소와 회복 기능, 게임의 결과를 나타내는 GUI, 마지막으로 모래바람과 효과 사운드가 재생되도록 구현해 보겠습니다.

생수병 모델 스크립트 작성하기

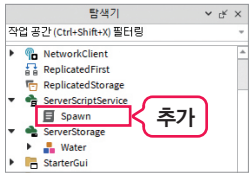
플레이어가 체력을 회복해 생명을 연장하기 위해서는 생수 아이템을 터치해야 합니다. 이번 스크립트에는 생수 아이템의 복제와 삭제, 고정, 용접 기능을 작성해 보겠습니다. 먼저 [탐색기] 창의 서버 전용 오브젝트를 보관하는 [ServerStorage]로 생수병 모델을 옮겨 놓습니다. 그리고 스크립트에서 복제된 생수병 모델을 구분해 관리할 수 있도록 [Workspace]에 [Folder]를 추가해 이름을 'WaterModels'로 변경하고 시작하겠습니다.



STEP 1 생수병 모델의 복제 및 삭제 스크립트

플레이어는 특정 지점에서 5초에 한 번씩 무작위로 생성됐다가 10초가 지나면 사라지는 생수병 모델을 찾아 다니게 됩니다. 일정한 주기로 생수병 모델이 복제되었다가 삭제되도록 스크립트를 작성해 보겠습니다.

01 생수병 모델의 복제는 모든 플레이어에게 동일하게 적용되어야 하므로 [탐색기] 창의 [ServerScriptService]에 [Script]를 추가하고, 'Spawn'으로 이름을 변경합니다.



02 [탐색기] 창에서 [Spawn] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 **생수병 모델 복제하기** Spawn

```
1  local WaterSpawner = game.Workspace.WaterSpawner:GetChildren()
2  local Water = game.ServerStorage.Water
3
4  while wait(5) do
5      local num = math.random(1, #WaterSpawner)
6      local clone = Water:Clone()
7
8      clone.Parent = game.Workspace.WaterModels
9
10     local X = WaterSpawn[num].Position.X
11     local Y = WaterSpawn[num].Position.Y
12     local Z = WaterSpawn[num].Position.Z
13
14     clone:MoveTo(Vector3.new(X,Y,Z))
15 end
```

코드 설명

1: GetChildren() 함수를 이용하여 생수병이 무작위로 나타날 지점을 WaterSpawner 변수에 테이블 형식으로 저장합니다.

NOTE GetChildren() 함수는 지정한 범위의 모든 하위 오브젝트를 테이블 형식으로 추출하는 함수입니다. [Spawn] 스크립트에서는 [WaterSpawner] 폴더 안에 있는 모든 오브젝트, 즉 생수병 모델의 생성 지점 8개를 추출합니다.

2: [ServerStorage] 폴더에 있는 생수병 모델 [Water]를 Water 변수에 저장합니다.

4: while 반복문을 이용해 5초마다 코드를 반복하여 실행합니다.

5: 1부터 WaterSpawner 파트 개수의 숫자를 무작위로 num 변수에 저장합니다. 즉, 1부터 8까지의 숫자 중 하나를 무작위로 저장하게 됩니다.

NOTE # 연산자는 루아 언어에서 테이블의 크기(항목의 개수)를 구하는 연산자입니다. 1, 2, 3, … 등의 숫자를 지정하지 않고 # 연산자를 이용하면, 항목의 개수가 변경되더라도 코드를 수정하지 않아도 됩니다.

6: Water 모델을 복제해 변수 clone에 저장합니다.

NOTE Clone() 함수는 지정한 오브젝트를 복제합니다. 오브젝트가 가지고 있는 속성과 코드까지 모두 동일하게 복제할 수 있습니다.

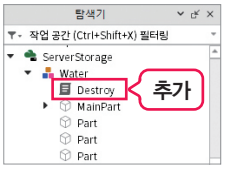
8: 복제된 모델을 [Workspace]의 [WaterModels] 폴더로 이동하여 관리합니다.

10~12: num 변수에 저장되는 n번째 무작위 지점의 위치 좌표를 X, Y, Z 변수에 저장합니다.

14: MoveTo() 함수를 이용해 복제된 Water 모델의 위치를 무작위 지점으로 이동시킵니다.

NOTE MoveTo() 함수는 [Position] 속성이 없는 모델의 위치를 변경하는 함수입니다. Vector3를 이용하여 이동하고 싶은 X, Y, Z 좌표를 설정할 수 있습니다.

03 생수병 모델이 복제되고 10초가 지나면 삭제될 수 있도록 [탐색기] 창의 [ServerStorage]에 있는 생수병 모델인 [Water]에 [Script]를 추가하고, 'Destroy'로 이름을 변경합니다.



04 [탐색기] 창에서 [Destroy] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 **생수병 모델 삭제하기** Destroy

```
1  local Water = script.Parent
2  wait(10)
3  Water:Destroy()
```

코드 설명

1: [Water] 모델을 Water 변수에 저장해 복제합니다.

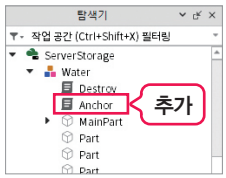
2~3: wait() 함수를 이용해 10초간 코드를 멈추고, Destroy() 함수를 이용해 [Water] 모델을 삭제합니다.

NOTE Destroy() 함수는 지정한 오브젝트를 게임에서 제거합니다.

STEP 2 생수병 모델의 고정 및 용접 스크립트

사막 맵은 모래 언덕으로 이루어져 있어 평평하지 않기 때문에 생성된 생수병 모델이 바닥에 고정되지 않고, 이리저리 굴러다니게 됩니다. 또한 여러 파트로 구성된 생수병 모델은 모델로 그룹화되어 있지만 게임을 실행했을 때는 각각의 파트가 따로따로 분리되므로, 이를 방지하기 위해 스크립트를 통해 생수병 모델을 고정하고 용접해 보겠습니다.

01 [탐색기] 창의 [ServerStorage]에 있는 [Water]에 [Script]를 추가하고, 'Anchor'로 이름을 변경합니다. 이 스크립트에서 생수병을 고정하면서 생수병의 반짝임 이펙트가 3.5초간 실행되도록 작성하겠습니다.



스크립트 생수병 모델 고정하기 Anchor

```
1  local Mainpart = script.Parent.MainPart
2
3  wait(0.5)
4  Mainpart.Anchored = true
5  wait(3)
6  Mainpart.Attachment.ParticleEmitter.Enabled = false
```

코드 설명

1: [Water] 모델의 MainPart를 MainPart 변수에 저장합니다.

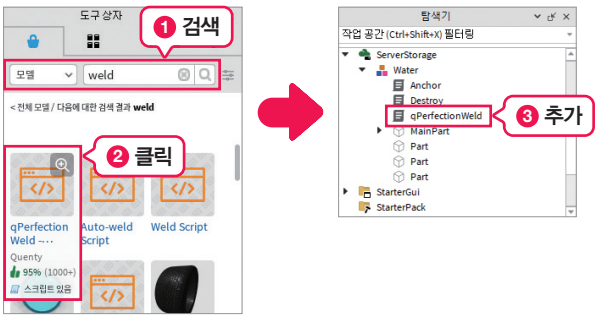
3: wait() 함수를 이용해 0.5초간 코드를 멈춰, 특정 지점에서 생수병 모델이 생성되고 자연스럽게 바닥에 고정될 수 있도록 설정합니다.

4: MainPart의 [Anchored] 속성을 true로 변경하여 생수병이 움직이지 않도록 고정합니다.

5: wait() 함수를 이용해 3초간 코드를 멈춥니다. 앞 코드의 0.5초까지 더해, 총 3.5초가 지나고 반짝임 이펙트의 실행을 종료합니다. 이 코드에서 대기하는 시간을 조정해 이펙트의 실행 횟수를 설정할 수 있습니다.

6: 반짝임 이펙트의 [Enabled] 속성을 false로 변경하여 코드의 실행이 종료되면 이펙트를 비활성화합니다.

02 생수병 모델의 용접 스크립트는 직접 작성하지 않고 [도구 상자]를 이용해 보겠습니다. [보기] 탭에서 [도구 상자]를 클릭하고, 드롭다운 목록의 [모델]에서 'weld(용접)'를 검색합니다. [qPerfectionWeld] 스크립트를 찾아 [탐색기] 창의 [ServerStorage] – [Water]에 추가합니다.



03 이제 [테스트] 탭에서 [플레이] 버튼을 눌러, 생수병 모델이 무작위로 특정 지점에서 생성되었다가 10초 후에 삭제되는지 확인합니다.

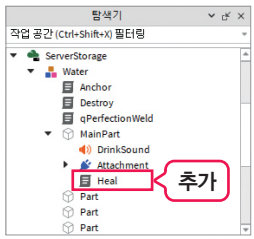
CHECKLIST

1	생수 아이템이 5초마다 특정 지점에서 무작위로 나타납니다.	
2	생수 아이템이 나타나고 10초가 지나면 사라집니다.	
3	생수 아이템이 반짝이며 나타납니다.	
4	생수 아이템이 땅을 굴러다니지 않고 한 지점에서 나타납니다.	

체력 스크립트 작성하기

플레이어의 체력이 게임의 성공과 실패를 결정합니다. 플레이어의 체력은 생수 아이템을 찾아 터치하면 회복되지만, 찾지 못하면 계속해서 데미지를 입습니다. 2초마다 −3씩 감소되었다가 생수병을 터치하면 회복되는 플레이어의 체력을 스크립트로 작성해 보겠습니다.

01 플레이어가 생수병 모델을 터치하면 체력이 회복되어야 하므로 [탐색기] 창의 [ServerStorage] – [Water] – [MainPart]에 [Script]를 추가하고, 'Heal'로 이름을 변경합니다.



02 [탐색기] 창에서 [Heal] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 플레이어의 체력 회복하기 Heal

```
1  local debounce = false
2
3  function Touch(Part)
4      local Player = game.Players:GetPlayerFromCharacter(Part.Parent)
5      if Player and debounce == false then
6          debounce = true
7          script.Parent.DrinkSound:Play()
```

```
8           wait(1)
9           Player.Character.Humanoid.Health += 10
10          script.Parent.Parent:Destroy()
11          debounce = false
12      end
13  end
14  script.Parent.Touched:Connect(Touch)
```

코드 설명

1: 연이어 실행되는 물 마시는 소리와 체력 회복 코드를 한 번만 실행하기 위해 true와 false로 지정할 수 있는 변수가 필요합니다. debounce 변수가 false이면 다음 코드를 진행하는 조건문을 작성하기 위해 debounce 변수를 생성하고 false로 지정합니다.

3: 충돌한 오브젝트의 정보를 받는 매개변수, Part를 이용해 Touch() 함수를 만듭니다.

4: GetPlayerFromCharacter() 함수를 이용하여 생수병 모델과 충돌한 플레이어의 정보를 Player 변수에 저장합니다.

5: 생수병 모델에 충돌한 오브젝트의 부모 객체가 플레이어의 캐릭터가 맞는지 확인하고, debounce 변수가 false인지 판단하는 조건문을 실행합니다.

6: debounce 변수를 true로 지정합니다.

7: 생수병 모델의 MainPart에 추가한 DrinkSound를 실행합니다.

8: wait() 함수를 이용해 물 마시는 소리가 끝날 때까지 1초간 기다립니다.

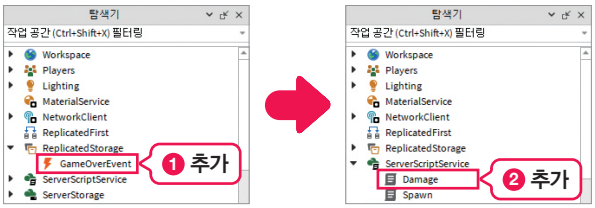
9: 충돌한 플레이어의 체력을 +10만큼 회복시킵니다.

10: Destroy() 함수를 이용해 복제된 생수병 모델을 게임에서 삭제합니다.

11: 다시 debounce 변수를 false로 지정합니다.

14: 다른 오브젝트가 MainPart와 충돌하면 Connect() 이벤트를 이용해 Touch() 함수를 실행합니다. 5번 행에서 정의한 조건문을 이용해 충돌한 오브젝트가 플레이어의 캐릭터인지를 판단한 다음, 맞으면 플레이어의 체력을 감소시킵니다.

03 플레이어의 체력은 계속해서 떨어지고, 제한 시간이 끝나거나 체력이 0이 되면 게임이 종료되어야 합니다. [탐색기] 창의 [ReplicatedStorage]에 [BindableEvent]를 추가해 'GameOverEvent'로 이름을 변경하고, [ServerScriptService]에 [Script]를 추가해 'Damage'로 이름을 변경합니다.



NOTE ReplicatedStorage는 특정한 상황에서만 플레이어에게 적용되어야 하는 객체를 저장할 때, ServerScriptService는 어떤 상황에도 변하지 않아야 하는 게임의 전반적인 규칙을 적용할 때 필요한 탐색기 항목입니다. 따라서 ReplicatedStorage에는 게임의 종료 신호를 받기 위해, ServerScriptService에는 플레이어의 체력에 연속적으로 데미지를 입히기 위해 스크립트를 추가합니다.

여기서 잠깐 **BindableEvent 알아보기**

BindableEvent는 동일한 스크립트 간의 신호를 전달하는 함수입니다. BindableEvent의 경우, 서버와 클라이언트간의 통신이 불가능하다는 점이 RemotEvent와 다른 점입니다. 즉, 클라이언트(Local Script)와 클라이언트(Local Script) 간의 통신, 서버(Script)와 서버(Script) 간의 통신만 가능합니다.

스크립트에서 이벤트를 이용해 신호를 보낼 때는 Fire()를 사용하고, 신호를 받을 때는 Event:Connect()를 사용합니다.

서버 → 서버로 신호를 보내는 경우

서버의 [Script] : 신호 보내기

```
BindableEvent:Fire()
```

서버의 [Script] : 신호 받기

```
BindableEvent.Event:Connect(function()
    -- 바인더블 이벤트 신호를 받았을 때의 실행문
end)
```

04 [탐색기] 창에서 [Damage] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 **플레이어의 체력 감소시키기** Damage

```
1  local GameOverEvent = game.ReplicatedStorage.GameOverEvent
2
3  game.Players.PlayerAdded:Connect(function(Player)
4      Player.CharacterAdded:Connect(function(Character)
5          Character.Health.Disabled = true
6          while wait(2) do
7              Character.Humanoid.Health -= 3
8          end
9      end)
10 end)
11
```

```
12  GameOverEvent.Event:Connect(function()
13      script.Disabled = true
14  end)
```

코드 설명

1: [ReplicatedStorage] 폴더에 생성한 GameOverEvent(BindableEvent)를 변수 GameOverEvent에 저장합니다.

3: 플레이어가 게임에 접속하면 함수를 실행하는 동시에, 플레이어를 매개변수 Player에 저장합니다.

4: 플레이어의 캐릭터가 게임에 추가되면 함수를 실행하는 동시에, 캐릭터를 매개변수 Character에 저장합니다.

5: 플레이어의 체력을 회복시키는 스크립트 [Health]를 비활성화시킵니다.

6~8: while 문을 이용해 2초간 연속적으로 플레이어의 체력을 -3만큼 감소시킵니다.

12~14: 게임의 종료 신호인 GameOverEvent(BindableEvent)를 전달받으면 스크립트의 Disabled 속성을 true로 변경해 더 이상 체력을 감소시키지 않도록 합니다.

CHECKLIST

1	시간이 지날수록 플레이어의 체력이 감소합니다.	
2	생수 아이템을 찾아 터치하면 플레이어의 체력이 회복됩니다.	

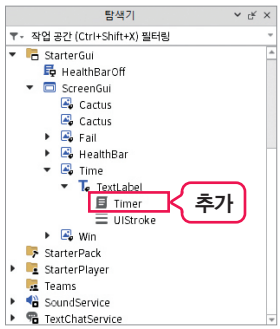
GUI 스크립트 작성하기

플레이어는 게임 화면에서 게임의 제한 시간과 체력을 확인하며 플레이합니다. 상황에 따라 실시간으로 GUI가 변화하면서 게임의 진행 상황을 확인할 수 있도록 스크립트를 작성해 봅시다.

STEP 1 제한 시간 GUI 스크립트

극한 생존 게임의 제한 시간은 1분 30초입니다. 따라서 게임 화면의 상단에 표시되는 제한 시간 GUI는 90초에서 시작해 0초로 줄어들어야 합니다. 플레이어가 주어진 90초 동안 사막 맵에서 생존하면 Win UI를 표시하고, 그 전에 플레이어의 체력이 0으로 떨어지면 Fail UI를 표시합니다. 마지막으로 게임이 종료되면 서버로 GameOverEvent 신호를 전송하는 것까지 스크립트로 작성해 보겠습니다.

01 [탐색기] 창에서 [StarterGui] – [ScreenGui] – [Time] – [TextLabel]에 [Script]를 추가하고, 'Timer'로 이름을 변경합니다.



02 [탐색기] 창에서 [Timer] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 **제한 시간 카운트하기** Timer

```
1  local Time = 90
2  local Win = script.Parent.Parent.Parent.Win
3  local Fail = script.Parent.Parent.Parent.Fail
4  local GameOverEvent = game.ReplicatedStorage.GameOverEvent
5  local Player = script.Parent.Parent.Parent.Parent.Parent
6
7  script.Parent.Text = Time.."초"
8
9  while wait(1) do
10     if Player.Character.Humanoid.Health <= 0 then
11         GameOverEvent:Fire()
12         Fail.Visible = true
13         break
14     elseif Time == 0 then
15         GameOverEvent:Fire()
16         Win.Visible = true
17         break
18     end
19     Time -= 1
20     script.Parent.Text = Time.."초"
21 end
```

코드 설명

1: 제한 시간을 카운트하기 위한 타이머 변수 Time을 만들어 1분 30초, 즉 90초로 지정합니다.

2~3: 앞에서 만든 Win GUI와 Fail GUI를 각각 변수 Win과 Fail에 저장합니다.

4: [ReplicatedStorage]에 있는 GameOverEvent(BindableEvent)를 GameOverEvent 변수에 저장합니다.

5: 해당 스크립트의 다섯 번째 부모 객체인 플레이어를 Player 변수에 저장합니다.

7: 타이머 변수(Time)의 남은 시간을 "초" 텍스트와 합쳐 TextLabel에 표시합니다.

9: while 문을 이용하여 1초마다 실행하는 반복문을 만듭니다.

10~12: 캐릭터의 체력이 0보다 작거나 같으면 [Damage] 스크립트로 GameOverEvent 신호를 보내 게임 화면에 Fail UI를 표시합니다.

13: break 함수를 이용해 반복문의 실행을 멈춰, 더 이상 타이머가 작동하지 않도록 합니다.

14~16: 캐릭터의 체력이 0보다 크고 타이머 변수(Time)가 0이 되면 [Damage] 스크립트로 GameOverEvent 신호를 보내 게임 화면에 Win UI를 표시합니다.

17: break 함수를 이용해 반복문의 실행을 멈춰, 더 이상 타이머가 작동하지 않도록 합니다.

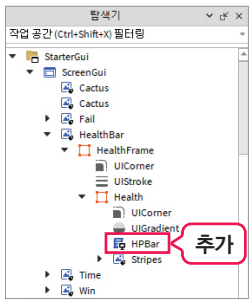
19: 타이머 변수(Time)에 -1을 적용해 게임의 플레이 시간을 감소시킵니다.

20: 감소된 게임의 플레이 시간을 TextLabel에 표시합니다.

STEP 2 체력 회복 GUI 스크립트

체력 회복 GUI에는 실시간으로 플레이어의 체력이 반영되어야 하고, 플레이어의 체력이 일정 수치 이하로 떨어졌을 때 막대의 색상이 바뀌면서 시간이 얼마 남지 않았음을 알려줘야 합니다.

01 체력 회복 GUI는 플레이어마다 각각 다르게 적용되어야 하므로 [탐색기] 창의 [StarterGui]에 [LocalScript]를 생성해 작성합니다. [탐색기] 창의 [StarterGui] – [ScreenGui] – [HealthBar] – [HealthFrame] – [Health]에 [LocalScript]를 추가하고, 'HPBar'로 이름을 변경합니다.



02 [탐색기] 창에서 [HPBar] 스크립트를 더블 클릭하여 스크립트 창을 띄운 후 다음과 같이 작성합니다.

스크립트 체력 막대 GUI 만들기 HPBar

```
1  local Health = script.Parent
2
3  while wait(0.2) do
4      local CurrentHealth = game.Players.LocalPlayer.Character.Humanoid.Health / 100
5
6  Health:TweenSize((UDim2.new(CurrentHealth,0,1,0)),Enum.EasingDirection.
       Out,Enum.EasingStyle.Quad,0.10)
7
8      if game.Players.LocalPlayer.Character.Humanoid.Health <= 40 then
9          Health.BackgroundColor3 = Color3.fromRGB(255, 170, 0)
10     else
11         Health.BackgroundColor3 = Color3.fromRGB(75, 255, 126)
12     end
13 end
```

코드 설명

1: 플레이어의 체력을 표시하는 GUI를 Health 변수에 저장합니다.

3: while 문을 이용하여 0.2초마다 실행하는 반복문을 만듭니다.

4: Health GUI는 0~1 사이의 값이고, 플레이어의 체력은 0~100 사이의 값이므로 둘의 단위를 맞춰 주기 위해 플레이어의 체력을 100으로 나눈 값을 변수 CurrentHealth에 대입합니다.

6: TweenSize() 함수를 이용해 CurrentHealth GUI의 크기를 변경합니다. GUI의 크기는 UDim2를 이용해 설정합니다.

NOTE TweenSize() 함수는 GUI의 크기를 자연스럽게 조정하는 데 이용하는 함수입니다. TweenSize() 함수의 매개변수는 TweenSize(GUI의 크기, EasingDirection, EasingStyle, Time)입니다. 여기에서 EasingDirection은 GUI가 표시되는 방향을 제어하는 값, EasingStyle은 GUI의 동작을 제어하는 값, Time은 GUI의 동작이 실행되는 시간을 의미합니다.

또한 UDim2의 값은 [X] 크기의 [Scale]과 [Offset], [Y] 크기의 [Scale]과 [Offset], 총 4개의 값으로 이뤄져 있습니다. [Scale]에 대한 내용은 책 225쪽에 있는 [여기서 잠깐]에서 다시 한 번 확인해 보세요.

8~9: 플레이어의 체력이 40 이하로 떨어지면 Health의 색을 '255, 170, 0(주황색)'으로 변경합니다.

10~11: 플레이어의 체력이 40보다 커지면 Health의 색을 '75, 255, 126(초록색)'으로 변경합니다.

여기서 잠깐 Color3란?

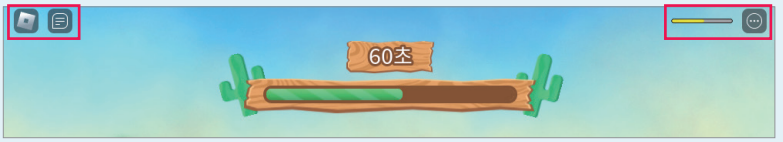
Color3는 RGB(빛의 삼원색 Red, Green, Blue)를 사용하여 색상을 설정하는 구성 요소입니다. 세 가지 색상의 값이 숫자 0~255 사이에서 설정됩니다. 색상을 설정할 때는 직접 RGB 값을 입력할 수도 있지만, 팔레트에서 마우스로 클릭해 사용할 수도 있습니다.

03 이제 [테스트] 탭에서 [플레이] 버튼을 눌러 GUI와 스크립트의 기능이 잘 구현되는지 확인합니다.

NOTE 처음 게임을 시작하고 아바타가 스폰포인트에 나타날 때, 아바타 주위로 나타나는 푸른색의 둥근 보호막도 속성 설정을 통해 없애고 넣을 수 있습니다. [탐색기] 창의 [Workspace]에서 스폰포인트 파트인 [SpawnLocation]을 선택하고, [속성] 창의 [Forcefield(보호막)] – [Duration(지속)] 값을 '0'으로 설정하면 보호막 없이 게임을 시작합니다.

여기서 잠깐 **체력 회복 막대가 2개?**

로블록스 스튜디오에서 게임을 플레이하면 메뉴나 채팅, 플레이어 목록 등 기본적으로 화면에 표시되는 UI들이 있습니다. 우리가 만들고 있는 극한 생존 게임에도 화면 오른쪽 상단에 체력 회복 막대가 표시되어 있는 것을 볼 수 있습니다.



우리는 이미 화면 중앙에 체력 회복 막대를 만들어 표시했으므로, 이 작은 체력 회복 막대를 표시하지 않기 위해 별도의 스크립트를 작성하겠습니다. [탐색기] 창의 [StarterGui] 안에 [LocalScript]를 생성하고, 다음 코드를 작성하면 됩니다. 예제 소스의 [부록] 폴더에 있는 'HealthBarOff.lua' 파일을 참고해 보세요!

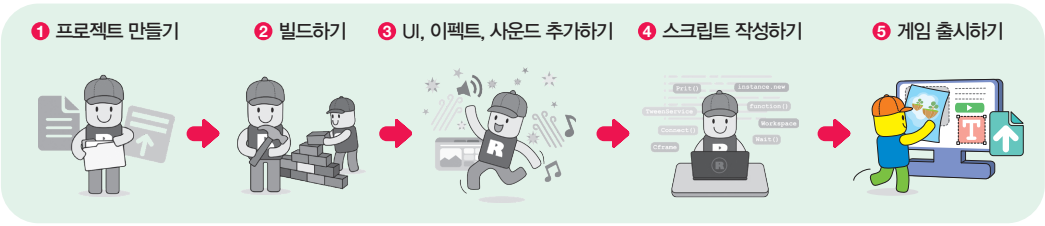
스크립트 **기본 체력 회복 막대 없애기** HealthBarOff

```
1  game:GetService("StarterGui"):SetCoreGuiEnabled(Enum.CoreGuiType.Health,
       false)
```

CHECKLIST

1	플레이어의 체력이 떨어지면 게임 화면에서 체력 회복 GUI의 색상이 주황색으로 바뀝니다.	
2	게임을 시작하는 동시에 제한 시간 GUI의 시간이 줄어듭니다.	
3	게임의 성공 또는 실패 GUI가 게임의 결과에 따라 알맞게 표시됩니다.	
4	게임이 종료되면 체력 회복 GUI의 체력이 더 이상 감소하지 않습니다.	

A-6 게임 출시하기



이제 게임의 출시만 남았습니다. 로블록스의 이용자들이 내가 만든 게임을 잘 찾을 수 있도록 썸네일을 추가해 출시해 보겠습니다.

01 로블록스 스튜디오의 [홈] 탭에서 [게임 설정] 버튼을 클릭해 [게임 설정] 대화 상자를 엽니다. 예제 소스의 [부록] 폴더에 있는 '아이콘.png' 파일과 '썸네일.png' 파일을 추가합니다.

02 세부 설정을 마치고 왼쪽 메뉴의 [권한] 탭에서 [플레이 가능 여부]를 공개로 바꿔 [저장]합니다. 이제 친구들이나 가족에게 내가 만든 게임의 링크를 공유해 함께 플레이해 볼 수 있습니다.

CHECKLIST

지금까지 배운 내용을 떠올리면서 결과물을 잘 만들었는지 스스로 체크해 보세요!

1	전체 게임의 콘셉트에 맞게 맵을 꾸몄습니다.	
2	게임을 플레이했을 때 선인장이 무너지지 않도록 고정했습니다.	
3	생수 아이템이 모래 바닥에서 무작위로 생성됐다가 일정 시간이 지나면 사라집니다.	
4	물 마시는 소리와 모래바람 소리에 알맞은 사운드를 찾아 적용했습니다.	
5	제한 시간 UI와 체력 회복 UI가 게임 화면에 알맞은 크기로 표시됩니다.	

찾아보기