우주선 찾기 게임

수업을 들으면서 데이터 블록과 추가 블록에 대한 활용이 거의 적어 아쉬움이 있었다. 실제 다양한 프로그램의 제작을 위해서는 데이터 블록과 추가 블록의 활용이 필요하다는 것을 친구로부터 들어 알고 있었기 때문이다. 그래서 이번에 준비한 프로젝트는 스크래치 사이트에서 찾은 플랫폼 형식의 탐험형 게임(https://scratch.mit.edu/projects/57571034/)의 논리를 간소화 및 추가하였다. 이를 통해 다소 어렵지만 학생들이 스스로 게임을 제작하는 경험을 얻고, 스크래치에 대한 더욱 깊은 이해가 뒤따를 것으로 여겨진다.

이 프로젝트의 제작 순서는 다음과 같다.

① 무대에 흰 배경을 남겨둔 채로 배경을 3개 더 추가한다.

② 추가한 각 무대에 맞는 음악을 3개 추가한다.

③ Ground 스프라이트를 추가한다.

④ Ground 스프라이트의 _r 모양과 _p모양을 적절하게 그린다.

⑤ Render 스프라이트를 추가하고, 렌더링할 모양들을 그린다.

⑥ Candle 스프라이트를 추가하고 모양을 2가지 만든다.

⑦ Render 스프라이트와 Candle 스프라이트, Ground 스프라이트가 렌더링을 할 때 유기적으로 연결되도록 논리를 구성한다.

⑧ Player Hitbox 스프라이트를 추가하고, 물리법칙을 비롯한 논리법칙을 설정한다.

⑨ Player Hitbox에 area의 전환과 관련된 논리를 설정하고 이를 무대에서 무대의 모양변화와 관련되게 논리를 만든다.

⑩ Player 스프라이트를 만들고, Player Hitbox를 따라가게 논리를 만든다.

⑪ Player 스프라이트에 손전등 모양을 추가하고, 이에 따른 조작 방법을 추가한다.

⑫ Player 스프라이트의 손전등 효과와 Candle 스프라이트 효과, 그 외의 밝기 효과등을 Render 스프라이트와 Player 스프라이트에서 설정한다.

⑬ Ufo 스프라이트를 추가하고, Player 스프라이트가 보내는 방송에 따라 사라지도록 스크립트를 작성한다.

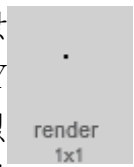
⑭ End 스프라이트를 추가하고, Ufo 스프라이트가 보내는 방송에 따라 나타나도록 스크립트를 작성한다.

이와 같이 간략하게 설명될 수 있다. 다만 아래에서 설명될 순서는 위의 제작순서와는 상이하게 진행될 것이다. 실제 제작 순서와 같게 스크립트들을 설명하려고 하면 한 단계에서 여러 개의 스프라이트들을 오가며 설명을 해야 하므로 혼란을 가져올 가능성이 크다. 그래서 연속성은 조금 떨어지지만

개별 스프라이트를 차례로 설명하고자 한다. 따라서 내가 원본 프로젝트를 보고 공부를 하면서 가장 중요성을 느꼈던 렌더링 과정과 연관된 스프라이트들을 먼저 설명할 것이다.(Render, Ground) 그 이후에는 Render 스프라이트 이외에 무대에 나타나는 배경 스프라이트(Candle, Ufo, End)를 설명할 것이다. 이는 렌더링 과정과 연결되어 있기 때문에 이해를 돕기 위함이다. 그 다음에는 물리 법칙과 관련된 Player Hitbox를, 그 다음에 Player 스프라이트, 마지막으로 무대에 관한 논리를 설명하고자 한다. 그래도 설명을 하면서 다른 스프라이트에서 보내는 방송과 관련되게 스크립트가 전개되는 경우 최대한 이에 대한 보충 설명을 하면서 진행할 것이다. 이로 인해 중복되는 내용이 다시 등장하기도 할 것이다. 또한 변수, 스프라이트 이름, 방송 등에서 유사한 이름이 등장할 수 있다. 그래서 이러할 때에는 최대한 구별이 되도록 설명할 것이다.

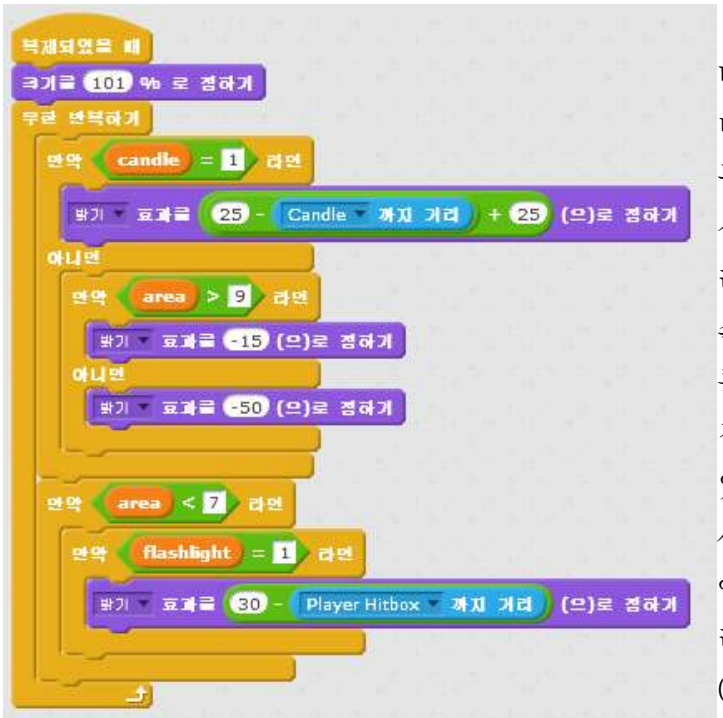
실제 플랫폼 형식의 게임을 제작할 때에는 먼저 **게임의 플랫폼**을 제작하여야 한다. 본 프로젝트에서는 이 플랫폼제작에 Ground, Render가 주로 관여한다.

기본적으로 **Render 스프라이트**는 플레이어가 올라서는 플랫폼을 꾸미는 스프라이트이다. Render 스프라이트의 논리는 다음과 같다.



이 논리 부분은 render라는 방송을 받았을 때 모양을 render로 바꾸고 X좌표와 Y좌표의 위치를 변경한다. 이는 이후에 있을 render 추가 블록의 실행에 필수적인 과정이다.

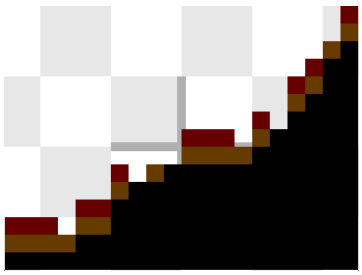
또 render 방송을 받았을 때 Render의 복제본을 삭제하여 화면을 정리하는 단계도 포함되어있다.



두 번째 논리는 다음과 같다. Render가 복제되었을 때 크기를 101%로 하여 빈틈없이 플랫폼의 배경이 형성되도록 한다. 그 밑의 논리는 이 프로젝트의 양초와 손전등으로 인한 **밝기 효과**를 나타낸 것이다. 이 프로젝트는 총 13개의 area가 있는데, 손전등은 6area까지 사용되고, 양초는 7~9area에 존재한다. 그래서 이 논리에 따르면 양초가 있는(candle=1) 경우 Render의 복제된 플랫폼들이 (25 - Candle까지의 거리)+25만큼 밝아진다. 양초에서

가까울수록 밝아지고, 멀어질수록 어둡게 설정된 것이다. 만약 양초가 없는 area라면 10area부터는 밝기 효과를 -15로 설정한다. 이때 무대에는 full moon이 있기 때문에 달빛의 효과를 낸 것이다. 그 외의 경우(area 1~6)에는 밝기를 -50으로 설정하여 복제된 플랫폼이 전부 검정색으로 보인다. 다만 이 때 손전등을 키면(flashlight=1) 밝기 효과를 30-Player Hitbox(캐릭터의 물리 효과 및 테두리 스프라이트)로 설정해 손전등과 가까울수록 밝아지게 설정하였다.

마지막은 **render 추가 블록**에 대한 정의를 담은 논리이다. **렌더링**은 render 방송을 받고 시작되는데, 이때 지정된(-228, -168)로 이동한다. 그다음 무대 위에 등장한다. 이 후 show 방송을 하는데 이때 무대는 잠시 흰 배경으로 바뀌고, 다른 스프라이트는 숨기기 설정이 되어 렌더링의 방해를 피한다. 또한 렌더링이 될 Ground는 show를 받았을 때 모양이 _r로 바뀐다. _r의 모양은 이와 유사하다.



이처럼 다른 스프라이트들이 숨겨지고 _r로 Ground가 바뀔 때 까지 기다린 후 렌더링이 시작된다. 먼저 x축 이동이다. 모양을 점으로 바꾼 후 오른쪽 방향을 본다. 이때 검은색에 닿으면 그 자리에 stone을 복제하고, 갈색에 닿으면 dirt를, 고동색에 닿으면 grass를 복제한다. 보라색에 닿으면 brick을 복제하고, 녹색에 닿으면 tree를 복제한다. 노란색이나 빨간색에 닿으면 각각 candle이나 beacon을 방송하는데 이는 각 스프라이트에서 추가 설명될 것이다.

(

, 

, 

, 

render의 stone, dirt, grass, brick, tree 모양. 이들이 복제된다)

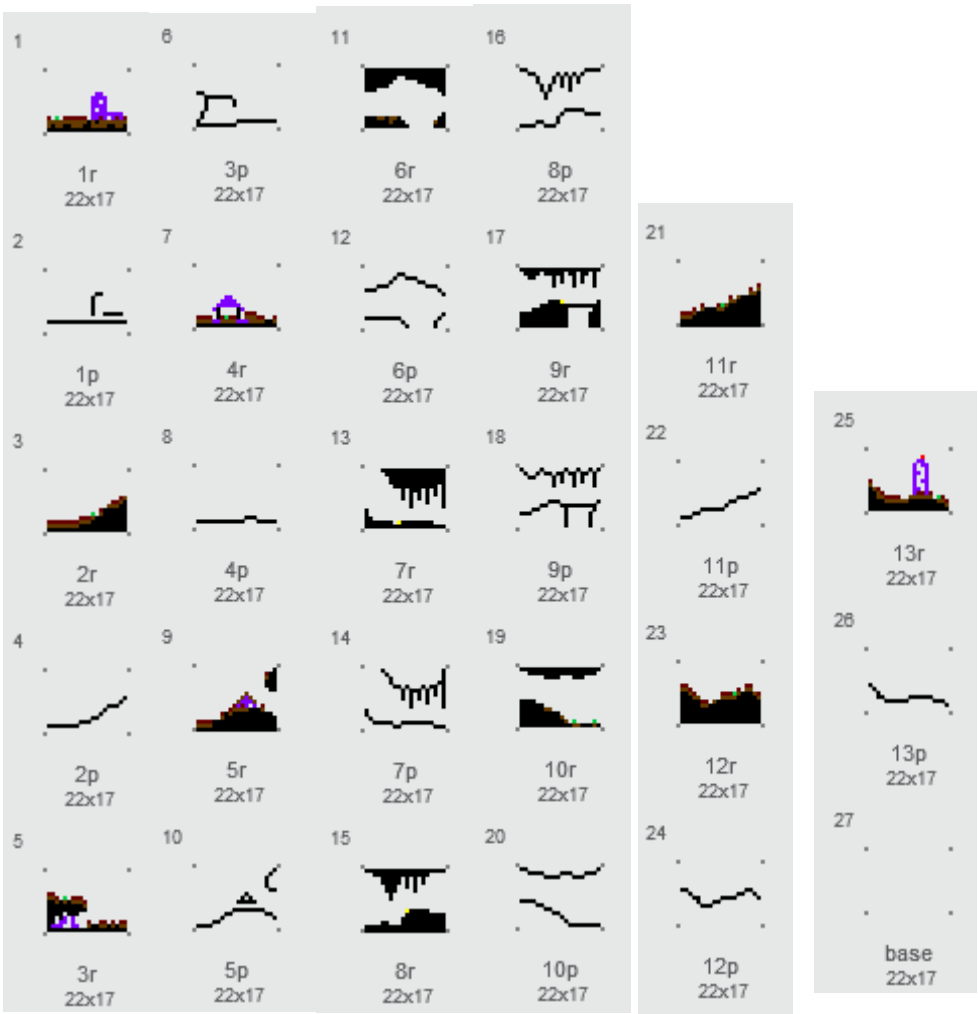
이렇게 복제가 이루어진 후에는 x좌표를 24만큼 바꾸고 다시 닿아있는 색에 따라 렌더링이 이루어지거나 아무 일이 없기도 한다. 이것을 총 20번 반복하는데 이는 스크래치의 전체 x좌표가 -240~240이기 때문이며, Ground _r의 제작과도 연관이 있다. 이는 Ground 스프라이트에서 다시 설명될 것이다. 어찌되었든 20번을 반복하면 x축을 한번 훑는다. 그리고는 x좌표를 다시 -228로 옮기고 y좌표를 24만큼 올려서 다시 x축을 훑을 준비를 한다. 이것을 15번 반복하는데 이는 스크래치의 y좌표가 -180~180인 것과 역시 Ground _r 의 제작과 연관되어 있다. 이와 같은 반복적 렌더링이 끝나면 스크래치 배경 전체를 한번 훑고, 접한 색상과 알맞은 모양을 복제하여 플랫폼을 꾸미게 된다. 이후에는 Render를 다시 숨기고 hide를 방송한다. 이 hide를 받으면 렌더링에 교란을 줄 수 있었던 무대와 플레이어 캐릭터 등을 다시 보이게 하여 게임을 재개한다.

두 번째로 설명할 스프라이트는 **Ground 스프라이트**다.

Ground의 논리는 단순하다. 녹색 깃발을 눌러 프로젝트가 시작되었을 때, render를 방송하고 기다린다. 그러면 위에서 설명한 것처럼 Render에서 show를 방송한다. show를 받으면 Ground의 크기를 24배 키운다. 그 후 그래픽 효과를 지우는데, 이는 hide 방송에서 설정된 반투명 효과를 지워 렌더링이 정상적으로 이루어질 수 있도록 하는 역할이다. 그 후 모양을 변수 area의 값과 r을 결합한 것(ex. 1r)으로 바꾼다. area변수는 화면이 넘어갈 때마다 1씩 늘어나도록 설정되어

있다. 이는 Player Hitbox 스프라이트에 설정되어 있으므로 그 때 추가적으로 설명할 것이다. r은 렌더링이 이루어지는 모양임을 나타낸 것이다. 이처럼 show의 논리를 수행하고 나면, 기다리고 있던 Render가 렌더링을 끝마치고 hide를 방송한다. hide를 받으면 반투명 효과를 100으로 설정하여 투명하게 하는 한편, 모양을 area의 값과 p를 결합한 것(ex. 1p)으로 바꾼다. p는 플랫폼을 뜻하는 것으로 실제 플레이어 캐릭터가 접촉하는 Ground의 모양이 된다. Player Hitbox에서 Ground를 통과하지 못하게 하는 논리가 있는데, 이는 (#area)p 모양의 Ground만 통과하지 못하게 함이다. 또한 Render가 hide를 방송하고 끝나면 start를 방송하는데, start를 받았을 때에는 (0,-12)로 이동한다. 이는 Ground의 모양이 24배 확대되었을 때 스크래치의 화면을 벗어나게 되어있기 때문인데, Ground를 화면 중앙에 정시키기 위함이다.

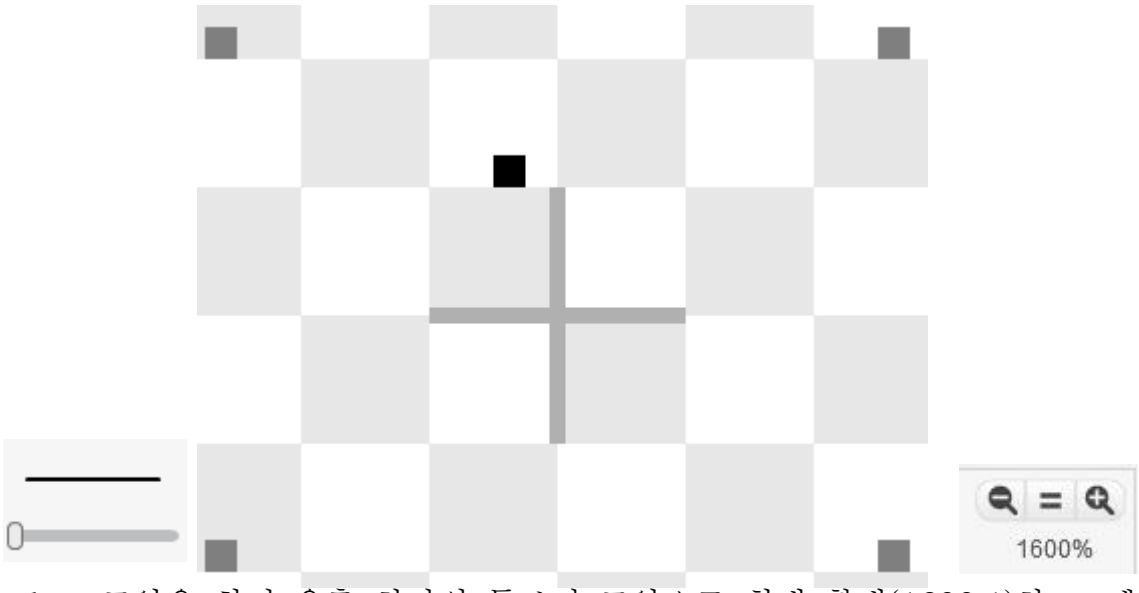
Ground에서 논리보다 중요한 것은 Ground를 직접 만드는 것일 것이다. 우선 Ground의 모양은 다음과 같다.



자세히 살펴보면 모양이 _r과 _p로 나뉜 것을 알 수 있다. _r은 렌더링이 이루어지는 모양인데, 위의 Render의 추가 블록에서 설정된 색상들을 모두 찾아 볼 수 있다. 검은색은 석벽이 되고, 갈색은 흙, 고동색은 풀, 보라색은

벽돌, 초록색은 나무가 된다. 7,8,9r에는 작게 한 픽셀의 노란색이 있는데, 나중에 Candle의 논리를 살펴보면, 이 자리에 Candle이 위치하게 된다. 13r의 빨간색 픽셀은 Ufo 스프라이트를 활성화 하는 역할을 하게 된다. 그 외의 _p모양들은 플레이어 캐릭터가 실제 접촉하는 Ground가 된다. Player Hitbox에 의해 통과할 수 없는 벽이 되는 것이다. 다만 이는 반투명효과 100으로 인해 실제 눈에는 보이지 않는다. 실제 게임화면에는 _r 모양에서 이루어진 렌더링의 결과물만이 플랫폼의 외형을 이룬다. base모양은 4개의 회색 사각형 내부에 모양을 그리면 24배 확대하고 (0,-12로 이동하였을 때 화면에 가득 차는 범위를 나타낸 것이다.

그렇다면 실제로 그려보도록 하겠다. 그릴 때에는 base를 기초로 한다.



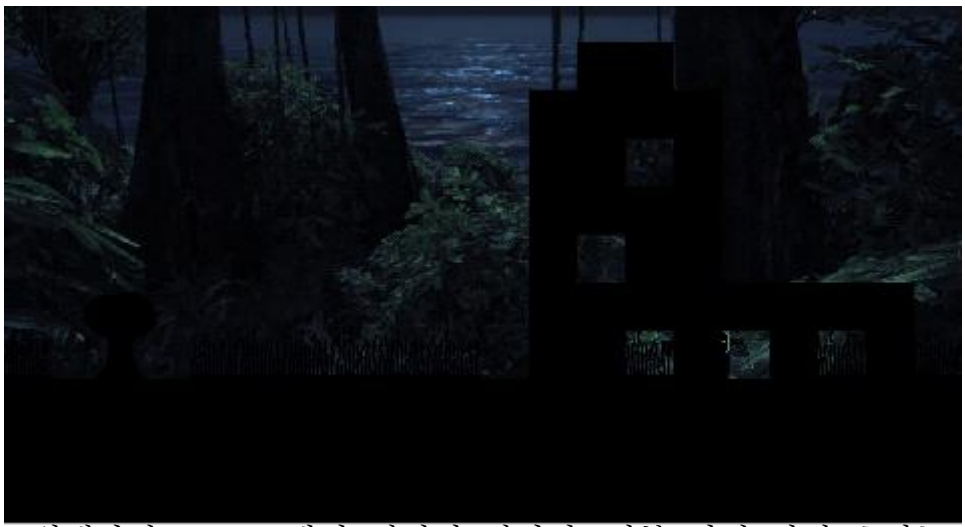
base모양을 화면 우측 하단의 돋보기 모양으로 최대 확대(1600%)하고, 펜 굵기를 최소로 하면 위와 같은 형태가 된다. 이 때 펜으로 점을 찍으면 회색 사각형 내부의 공간은 가로 20칸, 세로 15칸이 된다. 그래서 Render에서 가로로 20번을 반복하고, 위로 15번 이동한 것이다. 또한 Render 스프라이트의 위치가 (-228, -168)이라는 애매한 숫자로 설정된 것은 24배 확대된 한 픽셀의 가로, 세로가 24이기에 좌측 최하단의 사각형의 중심에 위치하도록 x, y좌표를 설정한 것이다.

이렇게 확대한 base의 내부에 지정된 색으로 모양을 만들어내면 _r로 이름을 붙여 렌더링 대상으로 만들 수 있고, 검정색만으로 플레이어가 올라설 공간을 설정하면 _p로 이름을 붙여 플랫폼으로 설정할 수 있다.

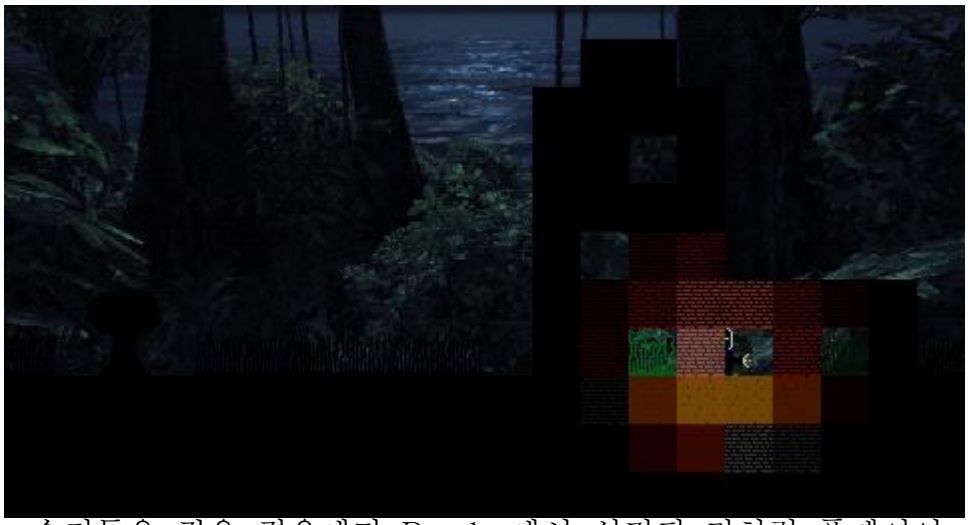
다음은 위의 논리들이 실제로 어떻게 작용되는지를 보여주기 위한 예시이다. 이것은 area 1의 r 모양이다.

이와 같은 1r모양이 설정되면 추가 블록 render를 거쳐

이렇게 바뀐다. 원래는 밝기효과 -50으로 설정되어 검정색으로 보여야 하나 전반적인 변화의 모습을 보이기 위해 임의로 밝기효과를 0으로 변경한 상태이다. 위의 검정색 픽셀은 전부 돌로 바뀌었고, 갈색 픽셀은 흙으로, 고동색 픽셀은 풀로 보라색 픽셀은 벽돌로, 녹색 픽셀은 나무로 바뀌었다.



원래라면 Render에서 설정된 것처럼, 기본 화면 밝기 효과는 -50으로 설정되어 모두 검정색으로 보인다.



손전등을 켰을 경우에만 Render에서 설정된 것처럼 플레이어 캐릭터 주위로 밝아지는 것을 볼 수 있다. 또한 플레이어 캐릭터도 약간 밝아지는데 이는 Player스프라이트에서 area 6까지는 밝기 -50을 기본 값으로 하고, 손전등을 켰을 때에는 밝기를 -20으로 정하게 하였기 때문이다.

또한 화면 중간에 플레이어 캐릭터가 석벽 뒤에 가려진 모습이 보이는데 이는 모양이 _p로 바뀐 후이기 때문이다.

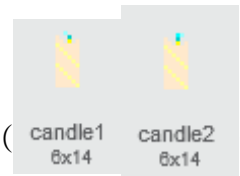
위의 장면에서 모양 1p는 이렇게 설정되어있다. 플레이어 캐릭터는 Player Hitbox에 설정된 논리에 의해 _p 모양의 검정색 벽들을 뚫을 수 없다. 위의 그림에서 석벽 뒤에 가려진 것(통과한 것)은 저 석벽이 모양 1p에 나타나있지 않기 때문이다. 만약 캐릭터가 가려진 위치에 1p의 검정 픽셀이 위치했다면 석벽 뒤에 위치할 수 없었을 것이다. 실제 게임 내에서는 반투명효과 100으로 인해 이것이 보이지는 않고 자연스럽게 진행된다.

이제는 플랫폼과 무대를 추가적으로 장식하는 스프라이트들이다. 여기에는 Candle, Ufo, End 스프라이트 등이 있다. 먼저 **Candle 스프라이트**를 소개하겠다.

Candle 스프라이트의 첫 번째 논리이다. 깃발을 눌러서 게임이 시작될 때 Ground 스프라이트가 render방송을 보낸다. 혹은 start방송으로 활성화된 Player Hitbox가 다음 area로 넘어가면서 방송되는 render방송에도 반응한다. 어떤 과정이건 이 render방송을 받았을 때 변수 candle을 0으로 정한다. 이 프로젝트에서 candle=0은 양초가 없음을, candle=1은 양초가 있음을 나타낸다. 매 렌더링마다 양초의 유무는 우선적으로 없음으로 초기화되고 Candle 스프라이트를 숨긴다.

그리고 렌더링 과정에서 Render 스프라이트가 노란색 픽셀과 닿는 경우 candle을 방송하게 되어있다. 이 candle 방송을 받았을 때 candle=1로 정한다. 즉 양초가 있음을 나타내는 것이다. 그 이후 노란색 픽셀에 닿아있는 Render 스프라이트의 위치로 이동하고 모양을 candle1로 바꾼 후 그 모습을 나타낸다. 그리고 candle2를 방송한다.

candle2방송을 받은 경우에는 1초마다 계속해서 모양을 바꾸도록 설정하는 논리이다.

(

Candle은 이 두 모양을 반복한다.)

실제로는 Candle 스프라이트가 어떻게 나타나게 되는지를 예시로 보이려 한다. 다음은 Ground 7r의 일부다.

검정색픽셀 다수와 노란색 픽셀 하나의 배경이다. 검정색 픽셀은 렌더링 과정에서 stone모양으로 복제될 것이며, Render가 노란색 픽셀에 닿았을 때

에는 candle 방송을 할 것이다.

Render가 candle 방송을 한 위치로 Candle 스프라이트가 이동하여 나타난 것을 알 수 있다. 또한 Render 스프라이트에 정의된 대로 렌더링과정에서 복제된 플랫폼 모양들은 Candle 스프라이트와의 거리에 따라 밝기가 달라지는 것이 보인다. 또한 플레이어 캐릭터의 밝기도 밝아진 것을 알 수 있는데, 이는 Render 스프라이트나 Candle 스프라이트에서 정의된 것이 아니라 Player 스프라이트에서 정의된 것이다. 이는 뒤에서 다시 언급할 것이다.

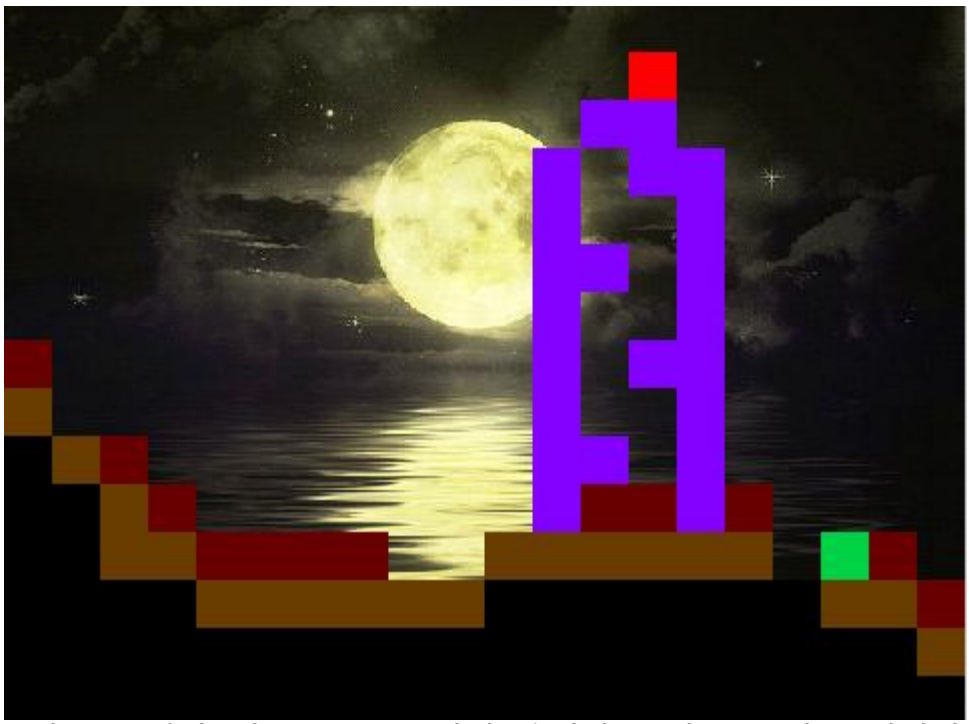
또 다른 배경을 장식하는 스프라이트는 **Ufo 스프라이트**이다. 이 스프라이트의 논리는 다음과 같다.

기본적으로 이 스프라이트는 대부분의 area에서 숨기기 상태이다. 먼저 그래픽 효과를 지운다. 이 스프라이트는 game over방송을 받았을 때, 반투명, 픽셀화 효과가 적용되는데 이를 복구하기 위함이다. 또한 이 스프라이트는 오직 area 13에서만 나타난다. 그래서 area 13의 기본 밝기 설정인 -15를 적용하였다. 그리고 (37, 139)의 좌표설정에도 나름의 이유가 있다. area 13에서 Player Hirbox의 논리에 의해 Player 스프라이트가 공중으로 이동을 하게 되는데, 위의 좌표에 Ufo 스프라이트가 있는 경우 자연스럽게 Ufo로 이동한 것처럼 보이기 때문이다. 마지막으로 이 스프라이트에는 9개의 모양이 있다. 이것이 0.2초 간격으로 돌아가면서 Ufo가 회전하는 느낌을 받게 하고자 함이다.

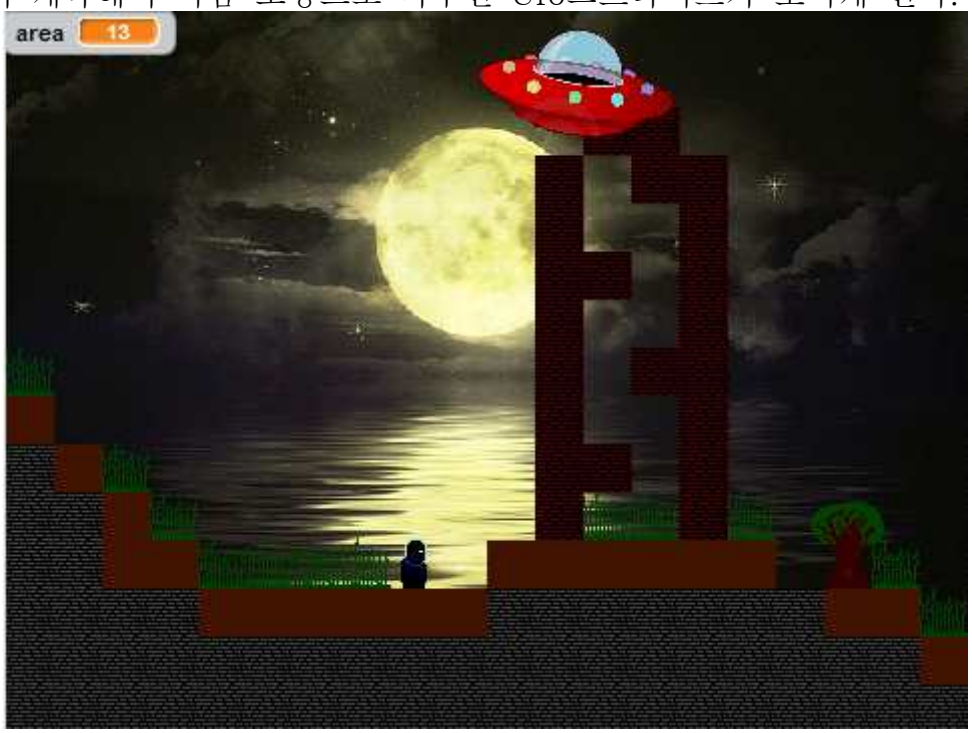
Ufo방송은 Render가 렌더링 과정 중에 빨간 픽셀에 닿을 경우 방송된다. 빨간 픽셀은 13r에만 존재하므로 이 스프라이트는 area 13에서만 나타난다. area 13에서 Player Hitbox의 x좌표가 10을 넘은 경우 end 방송을 하게 되는데, 이 end방송을 받았을 때, 이 스프라이트의 순서를 맨 앞으로 두어 Ufo가 Player 스프라이트를 가릴 수 있도록 하였다.

이후 Player 스프라이트는 투명화 되어 사라지는데, 이때 game over 방송을 한다. 이 방송을 받으면 1초 기다린 후 투명화, 픽셀화 되어 자연스럽게 사라진다.

다음은 Ground 13r의 모양이다.

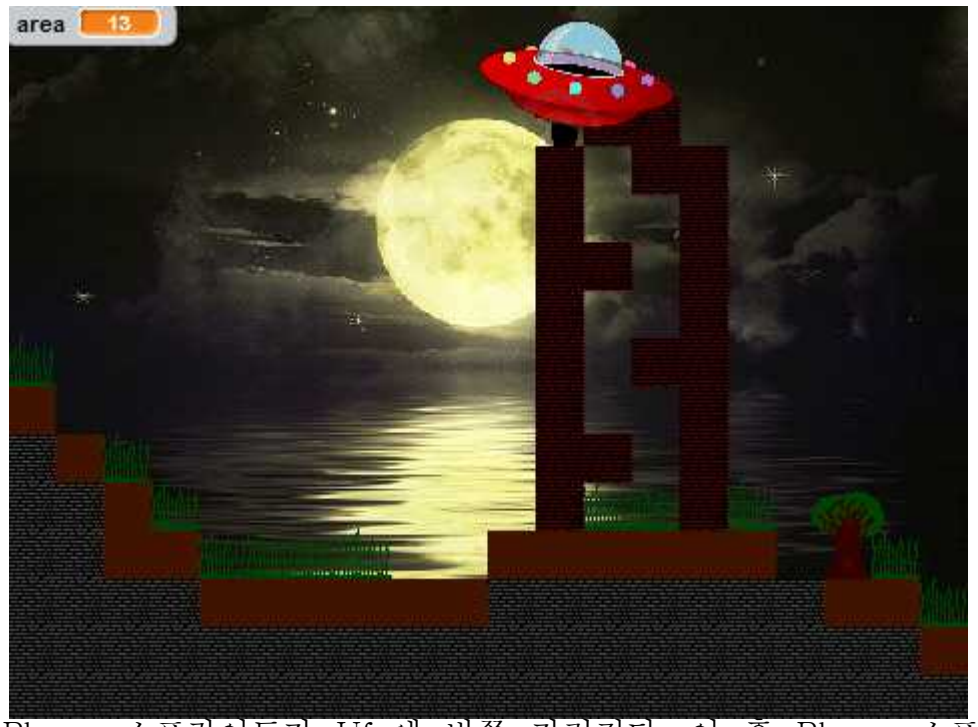


위 그림에서 이 프로젝트 전체 유일한 빨간 픽셀이 등장한다. Render가 렌더링 과정 중 빨간 픽셀에 닿으면 Ufo 방송을 하게 되고, 지정된 좌표에서 계속해서 다음 모양으로 바꾸던 Ufo스프라이트가 보이게 된다.



실제 area 13의 모습이다. area 10부터는 Render에서 정의된 밝기 효과

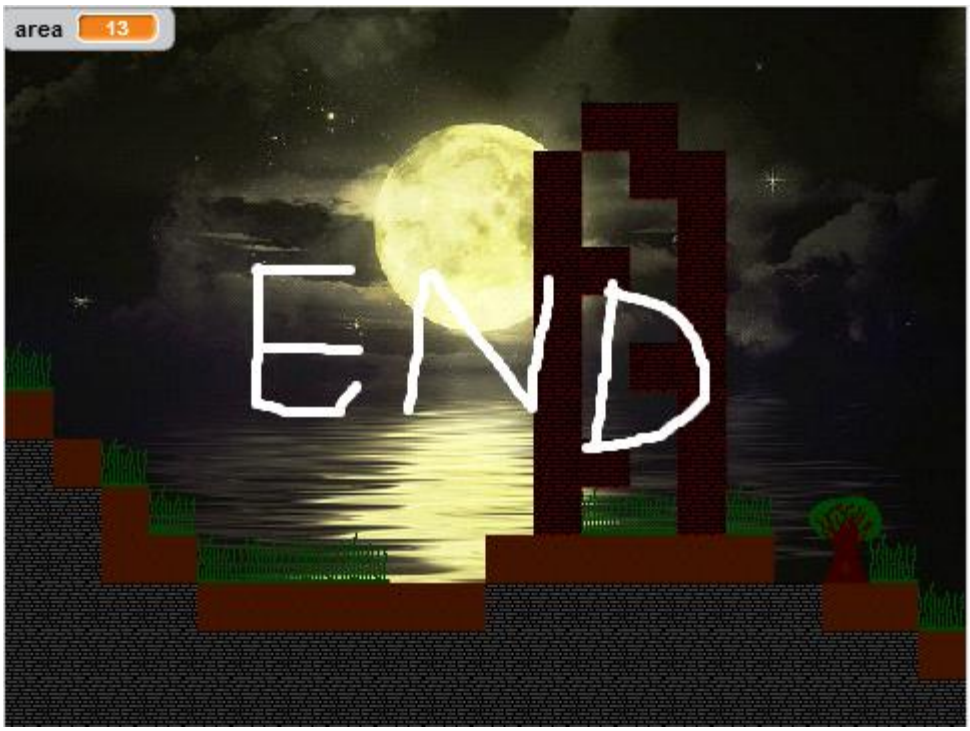
에 따라 모든 복제된 플랫폼들에 -15의 밝기 효과가 적용된다. 또한 캐릭터에도 Player스프라이트의 논리에 의해 밝기가 -15로 설정된다 특히 area 13에서는 Player Hitbox의 x좌표가 10을 넘어가면 end를 방송한다. Player Hitbox는 end 방송을 받으면 Y좌표<-65가 되는 순간 6초 동안 (40, 120)으로 이동하게 설정되어 있다. 캐릭터의 점프가 끝난 이후에 이동되게 하려는 목적이다. Player Hitbox의 논리에 의해 이동이 끝난 후에는 다음과 같은 모습이 된다.



Player 스프라이트가 Ufo에 반쯤 가려진다. 이 후 Player 스프라이트의 논리에 의해 Player 스프라이트가 투명화 되면서 게임이 끝이 난다.

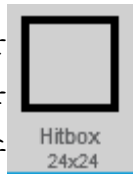
마지막으로 배경을 장식하는 스프라이트는 **End 스프라이트**이다.

이 스프라이트의 논리는 단순하다. 깃발을 눌러 게임이 시작될 때, 반투명 효과를 100으로 설정하는 한편, 숨기기를 통해 게임진행에 그 어떤 영향을 주지 않도록 한다. 그리고 Player 스프라이트가 투명화가 완료되면 방송되는 game over 방송을 받았을 때 다시 보이기를 통해서 게임 화면에 나타난다. 다만 3초를 기다린 후, 반투명효과를 음수로 설정해 점차 글자가 드러나는 형식이다.



Player 스프라이트가 Ufo에 반쯤 가려진 후, Player 스프라이트와 Ufo 스프라이트가 완전히 투명화 되면 화면상에 위와 같이 End 글자가 천천히 나타나게 된다.

그 다음으로는 실제로 움직이는 플레이어 캐릭터의 물리 법칙을 설정해야 한다. 캐릭터가 움직이는 **물리법칙**을 결정하는 것은 **Player Hitbox 스프라이트**가 관여한다. 이 물리 법칙을 Player 스프라이트가 아닌 개별 스프라이트로 설정한 까닭은, Player Hitbox처럼 사각형 형태의 스프라이트여야 물리 법칙의 판정이 정확하기 때문이다.



Player Hitbox 스프라이트를 자세히 살펴보기 위해서는 이 스프라이트에 사용되는 변수들을 알아야 한다.

area 변수는 현재 몇 번째 area인지 나타내는 변수이고, ground 변수는 Player Hitbox가 Ground 스프라이트에 닿아 있는지(=1), 안 닿아있는지(=0) 확인하는 변수이다. speed변수는 이 스프라이트의 기본 이동 속도를 의미하고, xV와 yV는 각각 x축과 y축의 좌표 변화 속도를 나타내는 변수다.

그렇다면 이제 Player Hitbox의 세부 논리들을 살펴볼 시간이다.

기본적으로 게임이 시작되면 area변수를 1로 설정하고, 이 변수를 화면상에 보이도록 설정한다. 이 변수에 따라 Render 스프라이트의 밝기 효과, Player 스프라이트의 손전등 효과, 무대 전환, Ground의 모양 변화 등이 이루어진다.

또한 Render 과정에서 show 방송을 받으면 숨기고, hide 방송을 받으면 다시 보이는 과정을 통해 렌더링 과정에 혼란을 피하도록 설계되었다.

이 스프라이트의 핵심 논리는 이 부분이다. 게임이 시작되었을 때, Ground 스프라이트에서 보내는 start 방송을 받으면 실행되는 이 논리는 2가지의 추가블록의 내용을 실행하도록 되어있다. 각 추가블록의 세부 내용은 다음과 같다.

먼저 살펴볼 것은 reset 추가블록이다. 게임을 새로 시작할 때 마다 xV와 yV를 초기화 시키고, 이 스프라이트의 기본 속도를 0.5로 지정하였다. 이 speed에 의해 xV의 속도가 변화하게 된다. 또 이 스프라이트의 기본 위치를 (-200, -96)으로 이동시키고 오른쪽 방향을 보게 한다. 그 다음 반투명 효과를 100으로 지정하는데, 이는 이 스프라이트가 Player 스프라이트의 물리 효과를 담당하는 스프라이트이기 때문에 스크래치 상에는 존재해야 하나 눈에는 보이지 않을 필요가 있기 때문이다. 그래서 반투명 효과를 설정하고 보이기 논리를 넣은 것이다. 마지막으로는 내 우주선이 어디 갔지? 라는 말을 2초 동안 하게 하여 게임의 진행 목적을 암시하였다.

그 다음은 physics 추가블록이다. 이 추가블록은 이 스프라이트가 어떻게 움직이는지 결정하는 논리들이 들어있다.

이 추가블록은 크게 2부분으로 나뉜다. y축 이동을 담당하는 부분과, x축 이동을 담당하는 부분이 그것이다.

먼저 y축 이동을 살펴볼 것이다. 기본적으로 y좌표를 yV 변수값 만큼 지속적으로 바꾸게 설정하였다. 이 때, 이 스프라이트가 Ground 스프라이트에 닿았다면 ground라는 또 다른 추가블록의 논리를 실행하도록 되어있다.

이것이 그 내용이다. Ground 스프라이트에 닿았는가? 가 아니다 까지 반복하기라는 모호한 설명이나 이는 Ground 스프라이트와 떨어질 때 까지 반복한다는 의미이다. 이 때 y좌표를 0-(절댓값 yV/yV)만큼 바꾸도록 하였다. 즉 yV값이 양수이면(올라가는 중이면) y좌표를 1만큼 내리고, yV값이 음수이면(떨어지는 중이면) y좌표를 1만큼 올린다. 즉 Ground를 y축으로 통과할 수 없도록 하는 논리인 것이다. 게임의 진행이 렌더링이 끝난 이후에 계속 이어지는 만큼, 이 논리를 통해 Ground _p 모양의 플랫폼을 통과할 수 없도록 하는 것이다. 다만 이 논리가 yV 변수의 값을 바꾸지는 않는다.

이 ground 추가 블록의 시행이 끝나면 추가 설정이 들어간다. 이 때 yV<0.1이라면, 즉 하강을 해서 yV값이 0이나 음수인 상태에서 Ground에 닿은 경우 yV를 0으로 정하고, ground변수를 1로 정한다. 땅에 닿아있다는 표시이다. 만약 그렇지 않다면, 즉 yV값이 0보다 커서 위로 올라가는 중에 Ground에 닿았다면 yV를 -1로 정해서 밑으로 떨어지게 하고, ground변수의 값을 0으로 정해 공중에 떠있음을 나타낸다.

그리고 Ground에 닿아있지 않다면 yV를 -1만큼 계속 바꾸게 하여 공중에

서 점차 밑으로 떨어지게 하였다. 이 논리가 중력의 역할을 한다. 이 논리 덕에 점프를 하면 처음에 빠르게 올라가다가 최고점에 도달한 후 점차 빠르게 낙하하는 것이다.

만약 ground=1(땅에 붙어있고), 위쪽 화살표를 누른 경우에는 yV를 12로 정해서 y축의 위쪽으로 상승하게 한다. 바로 위에서 설정한 중력의 역할을 하는 논리 덕에 자연스럽게 점프를 한다. 점프를 하면 공중에 뜨므로 ground를 다시 0으로 정하는 한편, 점프할 때 힘찬 소리를 내도록 설정하였다.

나머지 절반은 x축 이동에 관한 내용이다. x좌표를 변수 xV만큼 바꾸게 한다. 이후에 wall 이라는 추가블록이 사용되는데 이는 y축 이동에서 다루었던 ground 추가블록과 유사한 논리를 담고 있다.

wall 추가블록 역시 Ground 스프라이트에서 떨어질 때 까지 반복하도록 설정되어있는데, 이번에는 x좌표의 변화다. 만약 xV가 양수라면(오른쪽으로 이동 중이라면) x좌표를 -1만큼 바꾸어서 왼쪽으로 보내고, 만약 xV가 음수라면(왼쪽으로 이동 중이라면) x좌표를 1만큼 바꾸어서 오른쪽으로 보낸다.

즉 Ground 스프라이트를 x축으로 통과하지 못하게 하는 것이다. wall, ground 추가블록을 통해서 Ground _p 모양이 진정한 플랫폼으로서의 역할을 다하게 되는 것이다.

이후에는 이동에 대한 내용이 담겨있다. 만약 왼쪽 화살표를 누르면 xV를 reset에서 설정한 speed*(-1)로 설정하여 왼쪽으로 이동시키는 한편, -90도(왼쪽)방향을 보게 한다. 반대로 오른쪽 화살표를 누르면 xV를 speed*1로 설정하여 오른쪽으로 이동하게 하는 한편, 90도(오른쪽)방향을 보게 한다. 이를 통해 Player 스프라이트의 보는 방향을 조절하게 된다. 이는 Player 스프라이트가 Player Hitbox방향을 보도록 설정한 것을 유효 활용한 것이다. 마지막으로 ground=1(땅)의 경우에는 xV를 xV*0.75로 정하여 지상 마찰력을 구현하였고, ground=0(공중)의 경우 xV를 xV*0.85로 정하여 공기마

찰력을 구현하였다.

physic 추가블록에서 설정된 물리법칙이 변수를 이용하는 이동 방식으로 설정된 것도 까닭이 있다. 이 변수는 저장되어 다음 area에도 연결되는데, 다음 area로 넘어갈 때, 점프를 해서 넘어가면 다음 area에서도 점프를 한 상태로 넘어간다. 이는 xV와 yV값이 area를 넘어갈 때에 따로 초기화 되지 않고 연계되기 때문이다. 이처럼 자연스러운 이동을 하기 위해서 변수를 활용한 이동을 프로그래밍하였다.

이번에는 어떻게 area가 넘어가는지 설정한 논리이다. 렌더링이 처음 끝난 후 방송되는 start를 받았을 때 실행되는 논리이다. area가 5까지는 오른쪽 끝으로 이동하면 한 area가 끝이 난다.

그래서 area<6일 때 x좌표>235이면(오른쪽 끝에 닿으면) area를 1만큼 바꾸고, render를 방송하고 기다린다. 그러면 Render 스프라이트에서 show방송을 하여 Ground 스프라이트를 _r모양으로 바꾸고, 렌더링을 한 후 hide 방송을 하여 Ground를 _p모양으로 바꾼다. 이 모든 과정이 끝날 때까지 기다린 후에 x좌표를 -220(왼쪽 끝)으로 정한다.

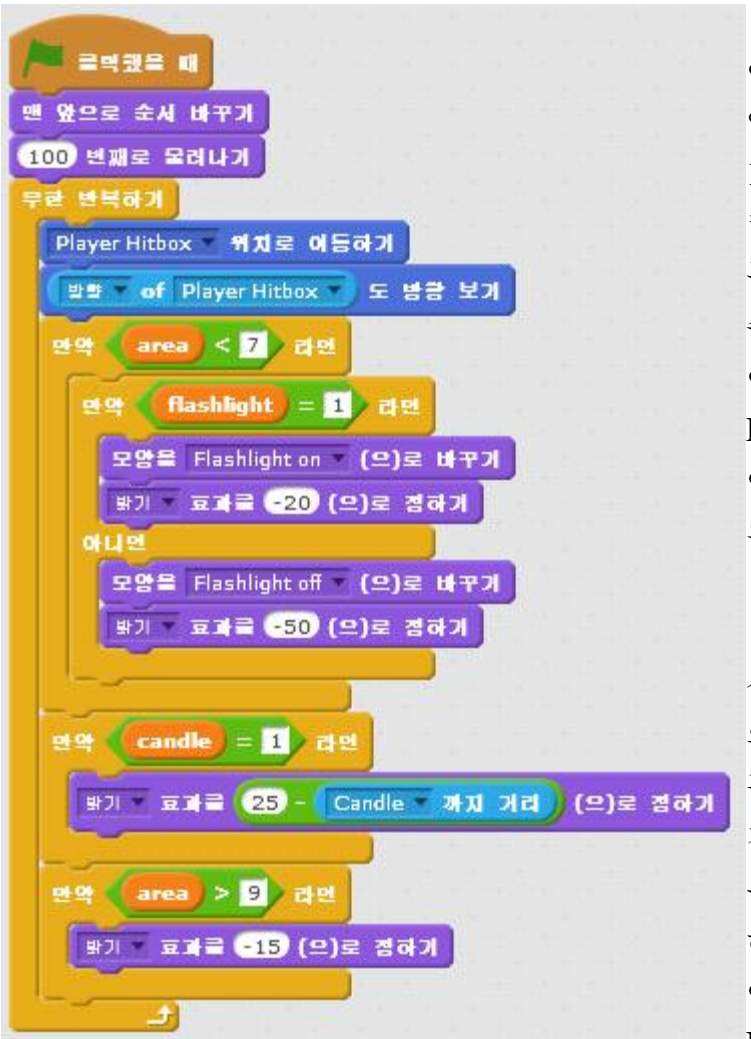
area 6에서는 동굴 바닥으로 떨어지게 되어있다. 그래서 y좌표<-175일 때 다음 area로 넘기고, 방송을 통해 렌더링을 기다린다. 이후 위에서 떨어져야 하므로 좌표를 (180, -220)으로 정한다. area 7부터는 손전등 효과를 사용하지 않기 때문에 손전등이 고장나버렸어... 라는 말을 하게 한다.

area 7부터는 다시 화면 제일 오른쪽에 닿았을 때 다음 area로 넘어가기 때문에 area<6과 같은 논리가 적용된다. 마지막 area 13에서는 플레이어 캐릭터가 자연스럽게 Ufo로 올라간다. 이를 위해서 x좌표가 10을 넘으면 end를 방송하게 한다.

이 end 방송을 받았을 때 y좌표<-65까지 기다린다. 즉 점프를 하고 내려와서 땅에 닿을 때 쯤 까지 기다린다. 이후에 이 스프라이트에 있는 다른 스크립트를 멈추게 하여 물리효과를 모두 멈추어 이동을 못하게 한다. 그리고 적당한 전송 효과음을 재생하면서 6초동안 (40, 120)으로 이동하여 Ufo에 반쯤 가려지게 되는 것으로 역할을 다한다.

결론적으로 Player Hitbox스프라이트는 캐릭터의 모든 물리 효과와 area 전환에 지대한 공헌을 하는 스프라이트이다.

반면 **Player 스프라이트**는 Player Hitbox 스프라이트에서 설정된 물리 법칙에 따라 움직이는 **아바타**이다.



이는 이 논리에 의해서이다. 게임이 시작되면 맨 앞으로 순서를 바꾼 후 100번째로 물러난다. 이는 렌더링 과정에서 복제된 플랫폼들의 뒤에 가려질 수 있게 하기 위함이다. 이후에는 반복적으로 Player Hitbox의 위치로 이동하고, Player Hitbox의 방향을 보게 한다.

즉 실제 게임에서는 보이지 않지만 실제로는 Player Hitbox와 일치된 움직임을 보이는 것이다. 앞서 말했던 것 처럼 이렇게 설정한 것은 물리효과가 더욱 정확하게 적용되게 하기 위함이다. 그 밑의 내용은 Player 스프라이트에 대한

밝기 효과이다. area6 까지는 손전등을 킬 수 있는데 손전등을 키면(flashlight=1) Flashlight on 모양으로 바꾸고 밝기를 -20으로 설정하며, 아니면 Flashlight off 모양으로 바꾸고 밝기 효과를 -50으로 설정한다.

(Player 스프라이트의 두 모양)

그리고 양초가 있는 경우(area 7~9) 밝기 효과를 25-Candle까지의 거리로 설정한다. 그리고 area 9부터는 다른 플랫폼들 처럼 밝기 효과를 -15로 정한다.

손전등을 키고 끄는 것은 이 논리가 결정한다. 기본적으로 flashlight(손전등의 유무)변수를 0으로(없음) 정한다. area6까지만 사용할 것이기에 area<7이라는 조건하에서 f키를 누를 때까지 기다린다. f키를 누르면 스위치를 키는 소리를 재생하고 flashlight를 1로 정한다. 그러면 위의 논리에 의해 Player스프라이트의 모양과 밝기가 바뀌고, 이 변수의 영향을 받는 Render 스프라이트에서도 Player Hitbox와의 거리에 따라 플랫폼들이 밝아진다. 그리고는 f키를 안 눌렀을 때 까지 기다리게 하는데, 이는 f키를 눌렀을 때 여러 번 중복 입력되어 손전등이 꺼졌다 켜지는 것을 반복하지 않도록 하기 위함이다.

이후에 다시 f키를 누르면 전원을 끄는 소리를 내며 flashlight를 0으로 정한다. 이에 따라 Player 스프라이트의 모양과 밝기가 바뀌고, 플랫폼들의 밝기가 다시 -50으로 정해진다.

하지만 area 7부터는 flashlight를 0으로 정하기만 하고 더 이상 키고 끌 수 없도록 설정하였다. 양초와 손전등이 같이 있는 경우 밝기 효과가 이상하게 적용되는 버그가 있어 아예 사용하지 못하도록 하였고, 게임 내에서는 손전등을 고장으로 인해 사용하지 못한다는 설정으로 하였다.

Player Hitbox가 end방송을 보내면 맨 앞으로 순서를 바꾼다. 이는 배경의 렌더링 된 벽돌 구조물에 가리지 않게 하기 위함이다. 그리고 10초를 기다려서 이 스프라이트가 Player Hitbox를 좇아 Ufo에 반쯤 가려질 때 까지 기다린 후, 드디어 집에 가겠군 이라고 2초 동안 말하게 한다. 그리고 반투명 효과를 2만큼 50번 주어서 점차 사라지는 느낌을 주며 Ufo에 타는 듯한 느낌을 준다. 그리고 game over 방송을 하는데 이 방송을 받으면 1초 후에 Ufo 스프라이트가 점차 사라지며, 3초 후에는 END글자가 화면에 나타난다.

이 외에도 이 스프라이트 역시 show를 받으면 숨고, hide를 받으면 보이게 하여 렌더링 과정에 혼란을 주지 않도록 설정되었다.

마지막으로 **무대**에 관한 설정이다. 무대에는 무대 화면의 전환과 BGM 설정이 담겨있다.

먼저 무대 화면의 전환에 대한 논리이다. 렌더링 과정의 시작 부분에서 show를 받았을 때에는 backdrop 기본 흰 바탕 무대로 바꾼다. 이는 렌더링 과정에서 혹시 모를 색 혼동으로 인한 렌더링 오류를 방지하기 위함이다. 렌더링이 다 끝나고 hide방송을 받으면 area 6까지는 jungle배경으로, 7~9는 cave배경으로, 그 외에는 full moon 배경으로 다시 바꾸게 하여 다시 무대를 불러 온다.

(

, 

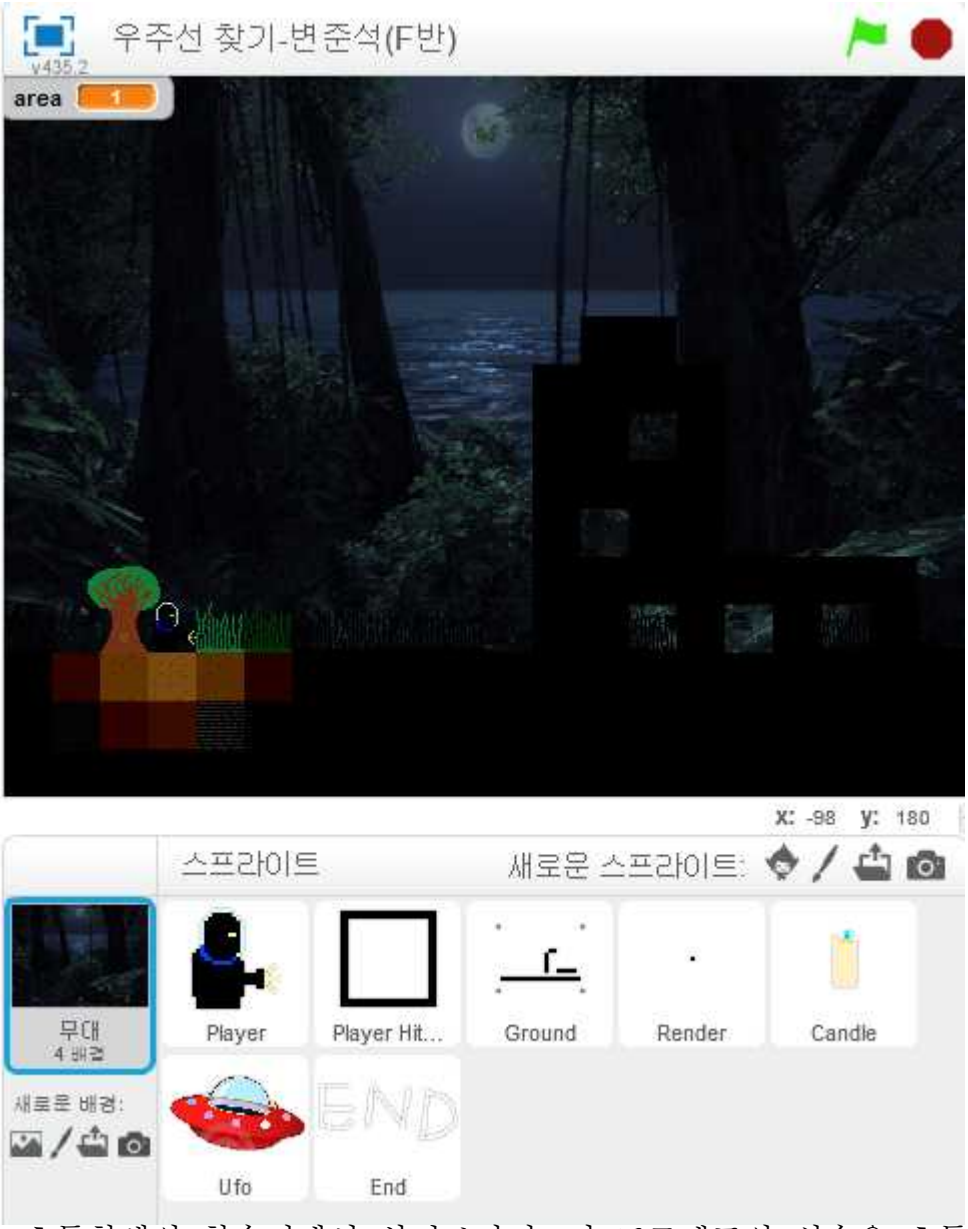
, 

, 

각 무대의 모습)

그 다음은 BGM관련 논리이다. 첫 번째 BGM은 area>6이 될 때 까지(다음 무대로 넘어갈 때 까지) 재생한 후 음량을 -2만큼 50번 내려서 소리를 다 내린 후 BGM을 정지시킨다. 그리고 다시 음량을 100%로 정한 후 다음 노래를 재생한다. 역시 area>9까지 기다린 후 점차 음량을 내린다. 그리고 다시 노래를 정지시킨 후 볼륨을 다시 100%로 정하고 마지막 노래를 재생한다.

이로서 https://scratch.mit.edu/projects/57571034/를 참조하여 만든 우주선 찾기 프로젝트의 모든 스프라이트들의 논리 설명이 끝이 났다. 처음 이 프로젝트를 접하였을 때에는 Render 스프라이트의 논리가 이해되지 않았고, Player Hitbox 스프라이트를 굳이 Player 스프라이트와 분리하였는지도 알 수 없었다. 또한 Player Hitbox에 들어있는 physics 추가블록과 관련된 논리들의 내용들도 처음에는 어렵게 느껴졌었다. 하지만 논리들을 개별적으로 삭제하고 첨가해가면서 이 논리들이 어떻게 작용하는 지 이해할 수 있었다. 나에게 기본 바탕 없이 플랫폼 게임을 만들어라 한다면 이제는 간단한 기본 형태의 플랫폼 게임은 손쉽게 만들 수 있을 것 같다. 더구나 이러한 게임 제작 형태의 스크래치 학습은 생소했던 데이터 블록과 추가 블록들을 충분히 사용하는 계기가 되었다. 다음에 현장에 나가서도 이러한 내용을 지도할 수 있을 것이라는 자신감이 드는 한편, 다음번 학기의 ICT 수업 때에도 스크래치를 가르친다면 데이터 블록과 추가 블록을 더욱 상세히 수업하셨으면 하는 바람이 있다. 스크래치 사이트에 올라와있는 대부분의 인기 게임 프로젝트들은 데이터 블록과 추가 블록 없이는 구동될 수 없는 구조였다.



초등학생의 학습면에서 살펴보자면, 이 프로젝트의 실습은 초등학생들에게 고난이도로 느껴질 수 있다. 하지만 초등학생들이 즐겨하는 여러 게임들의 수준을 고려할 때 이는 그렇게 높은 난이도는 아니다. 이 프로젝트는 어디까지나 스크래치 프로그램을 이용한 플랫폼 형식 게임의 기초를 다지기 위함이다. 무대에서는 일정 에리어마다 배경 화면을 변경하거나, 게임의 BGM을 설정하는 방법에 대한 연습을 할 수 있다. Player Htibox 스프라이트의 논리를 제작하면서 플랫폼 형식의 게임의 물리법칙을 설정하는 방법을 이해하여 자기가 원하는 조작법을 만들어 낼 수 있는 기회를 얻는다. Player 스프라이트는 게임 조작의 결과인 아바타의 모양을 결정하는 스프라이트다. 특히 flashlight를 키고 끄는 논리의 연습은 다른 것으로 응용하기 쉽다. 예를 들어 총을 쏠 수 있는 논리를 추가한 다음 특정 키를 눌렀을 때 총을 들거나 내리는 논리로 변형할 수 있다. Render 스프라이트는 렌더링 과정에 대한 이해를 도울 것이다. 특히 추가블록 render는 똑같은 논리를 그대로 다른 게임에도 적용할 수 있기 때문에 한 번 제대로 학습하면 이후에도 자신이 제작할 여러 게임에 쉽게 적용할 수 있다. Ground 스프라이트는 렌더에 필

요한 배경과 실제로 게임이 이루어지는 발판(플랫폼) 두 가지 모양의 변화에 대한 이해를 할 수 있다.

위의 무대와 4가지 스프라이트를 숙달하면 그 어떤 형식의 플랫폼 게임에도 응용할 수 있다. 위에서 떨어지는 다른 스프라이트에 닿으면 게임이 끝나는 형식이라거나, 화면 상의 동전을 먹으면 점수가 올라가는 형식이라거나, 화면 오른쪽에서 오는 적을 향해 총을 쏠 수 있는 형식의 게임 등 다양하게 응용이 가능하다. 이런 형식의 게임은 인터넷 상에 많이 있다. 이들의 작동 논리를 유추하여 학생들이 스크래치를 통해 스스로 만들어 내게 하려면 이 기초 논리의 학습은 필수적이다.

뿐만 아니라 이 프로젝트는 펜 블록을 제외한 모든 블록을 사용한다. 펜 블록이 정말 제한된 상황에서만 사용되는 것을 고려한다면 사실상 게임을 제작하는데 필요한 모든 블록들을 사용하는 것이다. 그래서 이 프로젝트를 스크래치 학습의 마지막 과정으로 사용한다면 기존에 배웠던 블록들에 대한 자연스러운 복습 및 심화 학습이 이루어 질 것이다.

마지막으로 이 프로젝트는 데이터 블록과 추가 블록의 활용이 많다. 데이터 블록과 추가 블록을 혼자서 처음부터 만들어 내는 것은 어렵다. 그러나 하나의 예시가 주어진다면 이를 활용해 다른 곳에 적용하는 것은 쉽다. 학생들에게 데이터 블록과 추가 블록 활용의 사례를 제시하는 것만으로도 이 프로젝트의 학습 가치는 높다.