피카츄 배구 물리학

피카츄 배구에 관한 모든 것

엑사바이트

Contents

서론 **3**

1장: 필드의 구조 **4**

1.1 필드 4
1.2 하늘 영역 5
1.3 땅 영역 5
1.4 네트 6

2장: 플레이어의 운동 **7**

2.1 플레이어 7
2.2 커맨드 7
2.3 조작키 8
2.4 상태 9
2.5 플레이어 이동 10
2.6 기초 게임 표기법 12

3장: 공의 운동 **13**

3.1 공 13
3.2 공의 이동 14
3.3 플레이어와 공의 충돌 14
3.4 정리 16
3.5 연습문제 16

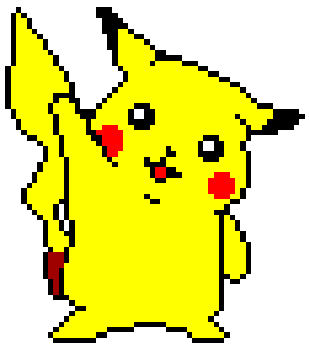
서론

피카츄 배구는 유구한 역사를 가진 컴퓨터실 게임이다. 아마 대부분의 사람이 피카츄 배구를 한 번 쯤은 해보았으리라 생각한다. 가벼운 게임이라고 생각할 수 있지만, 이 게임 뒤에는 엄청난 양의 연구가 숨겨져 있다! 썬더, 땅 긁기, 더블 어택, 속공, n단 콤보 등 여러 사람들에 의해 다양한 기술이 개발되어 있다. 지금도 기술 연구는 끊임없이 진행되고 있다.

이 문서에서는 피카츄 배구 게임을 물리학적 관점에서 바라보아 게임의 구조를 이해하는 것을 목표로 한다. 즉, 피카츄 배구의 내부 코드를 이해하여 공과 플레이어의 궤적을 직접 구할 수 있게 될 것이다. 더 나아가, 사용되고 있는 기술들의 원리를 수식으로 설명하도록 한다.

이 문서를 이해하기 위해서는 속도나 가속도와 같은 기초 물리학 지식과 중고등학교 수준의 수학이 필요하다. 그다지 어려운 내용이 아니므로 읽어나갈 수 있을 것이다. 이 문서의 내용을 직접 코드로 확인하고 싶다면 피카츄 배구 사이트의 **github 링크**를 참조하길 바란다. 주석과 함께 친절하게 설명되어 있다. 또한, 문서 내부 커맨드의 실행은 **피카츄 배구 서브 프로그램**에서 진행할 수 있다.

문서 작성에 도움을 준 **피카츄배구 온라인** 모든 분들께 감사를 전합니다.



1장: 필드의 구조

피카츄 배구 게임 필드, 플레이어, 공, 네트의 4가지 구성요소로 이루어져 있다. 첫 장에서는 이 필드의 구조에 대해 알아보도록 한다. (게임 자체와는 관련 없는 TMI가 많다.)

1.1 필드

필드는 게임에서 플레이어, 공, 네트를 제외한 모든 것, 즉 '바탕'을 뜻한다. 필드는 크게 하늘 영역과 땅 영역으로 나누어진다. Figure 1.1은 다른 물체를 제거한 피카츄 배구 필드이며, 크기는 432×304 픽셀이다.

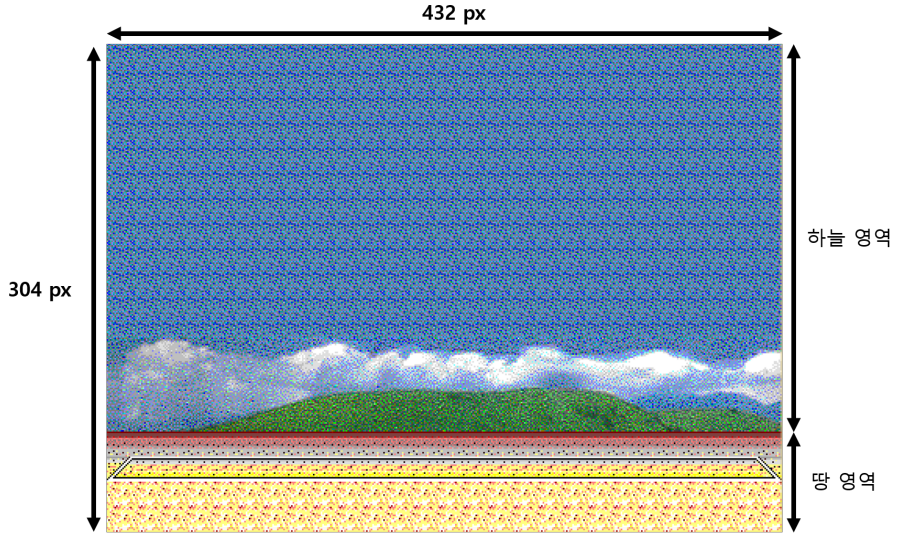


Figure 1.1 필드

앞으로 등장할 모든 좌표는 왼쪽 위를 $(0, 0)$으로 두고, 가로를 x축, 세로를 y축으로 한다. 예를 들어, 오른쪽 아래 꼭짓점의 좌표는 $(432, 304)$가 된다. 또한, 물체의 좌표를 나타낼 때 2가지 표기법을 사용하자. 대괄호 $[x, y]$는 물체의 왼쪽 위 점의 좌표를 뜻하고, 소괄호 (x, y)는 물체의 중심 좌표를 뜻한다.

1.2 하늘 영역

하늘 영역은 2가지 스프라이트로 구성된다. 하늘(SkyBlue) 스프라이트는 16×16 크기이며, 가로로 27개, 세로로 12개가 배열된다. ($16 \times 27 = 432$) 산(Mountain) 스프라이트는 432×64 크기이며, $[0, 188]$에 위치한다.

하늘 영역에는 총 10개의 구름이 돌아다닌다. (사람들이 잘 모르는 사실이다. 필자도 개발하면서 알았다.) 구름은 크기를 계속 바꿔가며 1부터 3까지의 랜덤한 속도로 움직인다. 게임 도중 배경을 보면 꿈틀거리며 움직이는 구름을 볼 수 있을 것이다. 구름 애니메이션이 이 게임에서 가장 복잡한 애니메이션이다.

1.3 땅 영역

땅 영역은 5가지 스프라이트로 구성된다. 빨간 땅(GroundRed), 노란 땅(GroundYellow), 경계선(GroundLine), 왼쪽 경계(GroundLine Leftmost), 오른쪽 경계(GroundLine Rightmost)이며, 모두 16×16 크기이다. 빨간 땅은 $[0, 248]$부터 가로로 27개 배열되며, 그 아래줄에 경계선 스프라이트들이 한 줄로 배열된다. 그 아래에는 노란 땅이 두 줄로 배열되어 총 4열의 땅 영역이 완성된다.

하늘 영역에 구름이 있다면, 땅 영역에는 파도가 있다. (이것도 모르는 사람이 많다.) 파도는 아래에서 위로 스프라이트가 올라오는 방식으로 표현되었다. 파도의 움직임은 구름의 움직임 다음으로 복잡한 애니메이션이다. 개발자가 구름과 파도에 진심이었다는 것을 느낄 수 있다.

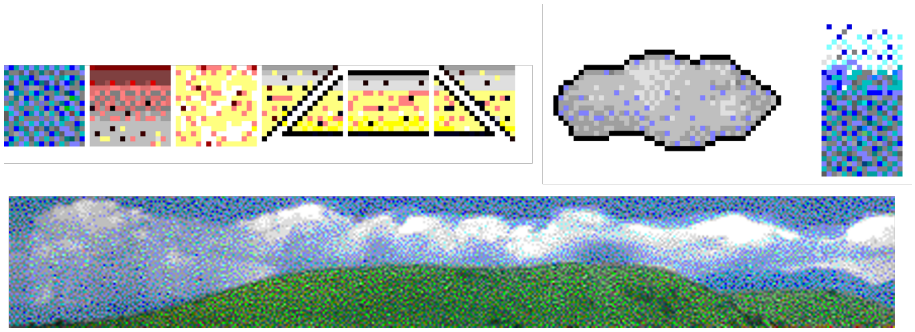


Figure 1.2 필드에 사용되는 스프라이트: 왼쪽 위부터 SkyBlue, GroundRed, GroundYellow, GroundLine Leftmost, GroundLine, GroundLine Rightmost, Cloud, Wave, Mountain

문제 1.3.1. 하늘과 땅 영역의 크기 및 차지하는 영역을 구하여라.

Proof. 땅 영역이 하늘 영역 위에 배열됨에 주의하자. 땅 영역은 $(0, 248)$부터 4줄이 배열되므로 $(16 \times 27 - 1, 248 + 16 \times 4 - 1) = (431, 311)$까지 덮는다. 하지만, 전체 필드의 크기는 432×304이므로, 땅 영역이 차지하는 영역은 $(0, 248)$부터 $(431, 303)$이다. 따라서 하늘 영역은 $(0, 0)$부터 $(431, 247)$까지의 영역을 차지하며, 하늘 영역과 땅 영역의 크기는 각각 432×248, 432×56이다. □

1.4 네트

네트는 전체 필드를 2개의 코트로 구분하는 물체이다. 네트는 2가지 스프라이트로 구성된다. 네트 꼭지(NetPillarTop)과 네트 기둥(NetPillar)이며 모두 8×8 크기이다. 네트 꼭지는 $[213, 176]$에 배열되고 그 아래에 12개의 네트 기둥 스프라이트가 배열된다. 네트는 필드 정중앙에 위치하지 않는데, 아래 문제에서 확인할 수 있다.



Figure 1.3 네트 스프라이트: 왼쪽부터 NetPillar, NetPillarTop

문제 1.4.1. 네트의 크기 및 차지하는 영역을 구하여라.

Proof. 네트는 $(213, 176)$부터 $(213 + 8 - 1, 176 + 13 \times 8 - 1) = (220, 279)$까지 8×104 크기의 영역을 차지한다. 이를 통해 네트가 정중앙에서 오른쪽으로 **1px** 떨어진 곳에 있다는 것을 확인할 수 있다. □

필드에 네트를 배치하면 다음과 같다.

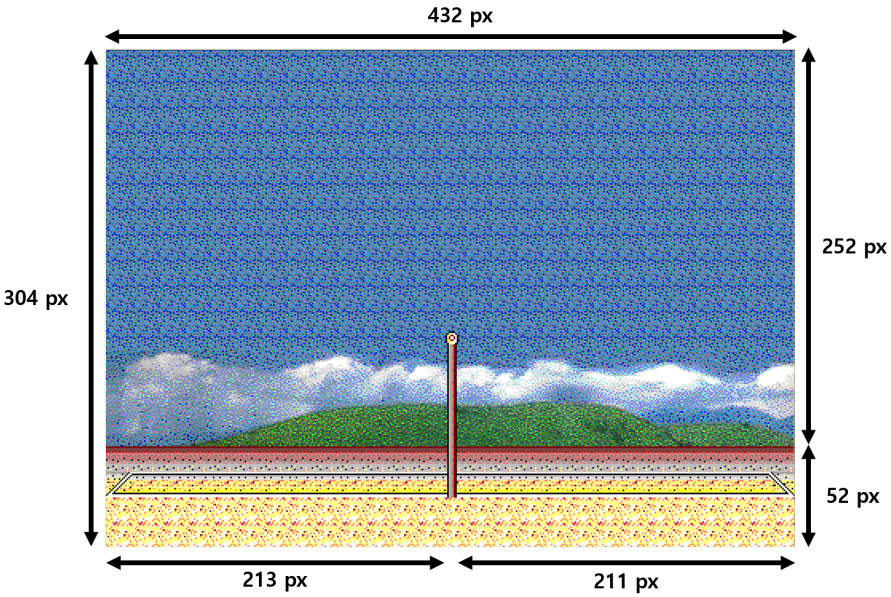


Figure 1.4 네트를 배치한 필드

2장: 플레이어의 운동

2장부터는 본격적인 게임 시스템에 대해 공부한다. 가장 기초적이면서 중요한 플레이어의 움직임에 대해 알아보도록 한다.

2.1 플레이어

플레이어란, 우리가 조종하는 피카츄를 말한다. 대전 게임이기 때문에 1p와 2p의 2개의 피카츄가 있다. 두 피카츄는 같은 스프라이트를 사용하며, 왼쪽의 플레이어를 1p, 오른쪽의 플레이어를 2p라고 한다. 스프라이트의 크기는 64×64이며, 플레이어의 위치를 나타낼 때는 항상 피카츄 중심의 좌표를 사용하도록 한다. Figure 2.5는 모든 피카츄 스프라이트로, 모두 64×64 크기이다. 충돌이 일어날 때도 항상 **정사각형**으로 간주되는 것에 유의하자.

플레이어는 게임이 시작될 때 항상 같은 위치에서 시작한다. 1p는 $(36, 244)$에서, 2p는 $(432 - 36, 244) = (396, 244)$에서 시작하게 되며, 좌우 대칭이란 것을 확인할 수 있다. 플레이어는 위치와 속도를 가진다. 2차원이므로 위치는 2차원 벡터로 표현되고, 속도는 y축 속도만이 필요한다. x축 움직임의 경우 단순히 위치에 상수를 더하고 빼는 식으로 움직인다. y축으로는 중력이 작용하는 등가속 운동을 한다.

2.2 커맨드

커맨드란, 플레이어를 조작하기 위한 명령을 말하며, **1프레임마다 실행된다.** 커맨드는 1p와 2p가 독립적으로 실행되며, 1p와 2p가 실행할 수 있는 커맨드는 상하좌우와 스파이크 총 5개로 동일하다. 이제부터 상하좌우의 커맨드를 **UDLR**(Up Down Left Right)로 표현하고, 스파이크를 **H**(Hyperball)로 표현하자. 이를 **기초 커맨드 표기법**이라 한다. 이제부터 커맨드를 하나씩 자세히 살펴보자.

L과 R은 좌우 이동을 실행하며, L과 R은 1프레임 당 플레이어를 6px 이동시킨다. 플레이어가 벽에 도달하게 되면, 더 이상 L과 R을 실행해도 벽을 넘어서 움직이지 못한다. 1p와 2p 모두 우선 순위는 L¿R이다. 즉, 왼쪽과 오른쪽 커맨드를 모두 실행하면 항상 왼쪽으로 이동한다.

U는 흔히 말하는 점프키로, 피카츄배구를 어렵게 만드는 가장 큰 요인이다. U는 플레이어의 y방향



Figure 2.5 모든 피카츄 스프라이트

속도를 -16으로 만들어서 점프를 시킨다. 점프를 해야만 공을 스파이크할 수 있기 때문에 점프하는 것은 필수적이지만, 점프를 하면 자신의 코트를 지킬 수 없기 때문에 수비에 매우 약해진다. 따라서 우리의 목표는 점프를 최소한으로 사용하면서 공격하는 것이다.

D는 아래 방향을 의미하며, 가장 적게 눌리지만 필수적인 키라고 할 수 있다. D가 유일하게 사용되는 곳은 스파이크이며, 스파이크 키와 D를 같이 눌렀을 경우 공이 아래를 향한다. U와 D를 같이 실행할 경우, 우선순위는 U가 높기 때문에 D는 씹힌다.

H는 스파이크 커맨드로, 피카츄배구의 모든 것은 H로부터 나온다고 해도 과언이 아니다. 고수 중에서도 H 없이 초보를 이길 수 있는 고수는 손에 꼽는다. H는 스파이크와 다이브의 2가지 역할을 한다. 공과 충돌했을 때 H를 누른다면 공은 Hyperball 상태로 바뀌고, 속도가 빨라지며 이를 스파이크(Powerhit)이라 한다. 점프하지 않은 상태로 L/R과 함께 H를 실행하면 앞으로 돌진하는 다이브(Dive)가 실행된다. (배구에서는 디그라고 한다.) **다이브는 순간적으로 x방향 속도를 높여주고, 작은 높이로 점프를 한다.** 다이브 후에는 움직일 수 없는 스턴 상태가 있다. 자세한 것은 뒤에서 살펴보자.

2.3 조작키

기존 피카츄 배구의 조작키는 1p와 2p가 구분되어 있다. 1p는 RDFG+V+Z, 2p는 화살표키+Enter이다. 1p의 RDFG는 WASD를 오른쪽으로 2칸 이동시킨 것이다. 중수부터는 알 것이라 생각되지만, 1p는 2p보다 조작키 수가 하나 더 많다! 아래에서 살펴보자.

조작키와 커맨드는 다르다! (중요!!!!!) 2p의 경우 앞선 커맨드와 모든 방향키가 일대일대응된다. 그러나, 1p의 경우 Key R-U, Key D-L, Key G-R, Key Z-H까지는 동일하지만, **Key F는 D+R의 커맨드를 실행하고, Key V가 D의 커맨드를 실행한다!** 즉, 1p에서 F키를 누르면 동시에 오른쪽으로

가게 된다. 이를 **F키 중첩 현상**이라 한다.

F키 중첩 현상을 확인할 수 있는 예시 2개를 살펴보자. 첫번째로, 1p에서 시작할 때 F키만 쭉 눌러보자. 분명 아래키를 눌렀는데 오른쪽으로 이동하는 것을 확인할 수 있을 것이다. 두번째로, 1p에서 아래 찍기를 F키를 이용하여 실행하면, 아래 대각 찍기가 나가는 것을 확인할 수 있다. 따라서 1p에서 2p와 같은 아래 찍기를 실행하려면 V키를 사용해야 한다. (V키가 2p의 아래 키에 대응된다.)

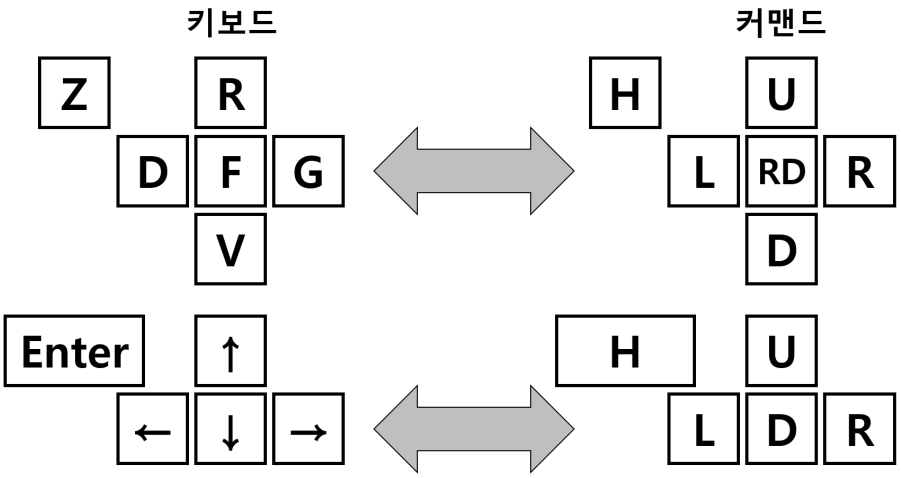


Figure 2.6 키보드와 커맨드

2.4 상태

플레이어는 항상 **상태(State)**를 가진다. 총 5가지의 상태를 가지며, 각각 Normal, Jump, Powerhit, Dive, Lie이다. (Win과 Lose까지 총 7가지로 볼 수도 있다.) Normal은 아무것도 하지 않은 기본 상태이다. Jump는 말 그대로 점프하여 발이 땅에 닿지 않는 상태, Powerhit는 스파이크 도중인 상태, Dive는 다이브 도중인 상태, Lie는 다이브 종료 후 움직이지 못하는 상태(스턴 상태)이다. 플레이어는 이 5가지 상태를 계속 전환해가는데, 아래는 상태 간의 전환 그래프이다.

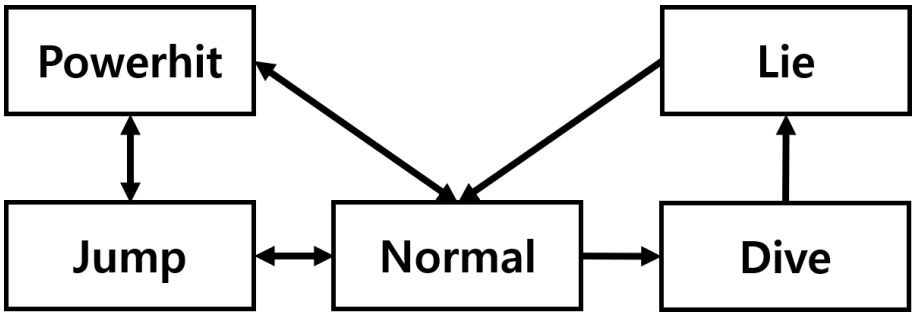


Figure 2.7 플레이어의 상태 전환

2.5 플레이어 이동

본격적인 내용에 들어가기 앞서, 여태까지의 내용을 정리하자. 플레이어는 1프레임 당 커맨드의 집합을 입력으로 가지고, 그에 따라서 움직인다. 또한, 플레이어는 상태를 가지며 진행에 따라 상태를 전환한다.

이제부터 이 장의 하이라이트인 플레이어 이동에 대해 살펴보자. 플레이어 이동 알고리즘은 0단계부터 4단계까지 총 5단계로 구성된다. 0단계에서는 다이브 후 Lie 상태일 경우 아무것도 하지 않고 이동을 종료한다. 구체적으로는 Lie 상태가 된지 5프레임 이내일 경우 종료한다. Lie 상태가 아니거나, Lie 상태가 된지 5프레임이 넘은 경우 1단계로 진행한다.

1단계는 x좌표 연산으로, 플레이어 상태가 Normal, Jump, Powerhit이라면 커맨드 방향으로 6만큼 움직인다. 플레이어 상태가 Dive라면, 이전에 입력했던 다이브 방향으로 8만큼 움직인다. (당연하게도 다이브 속도가 더 빠르다.) 움직였을 때의 x좌표가 코트를 벗어나면, 플레이어를 코트 끝으로 이동시킨다. 1p의 경우 $32 \leq x \leq 184$까지, 2p의 경우 $248 \leq x \leq 400$까지가 코트이며 이를 벗어날 수 없다. (네트는 비대칭인데 코트는 대칭인 것을 볼 수 있다.)

2단계는 y좌표 연산이다. 먼저, (플레이어 상태가 Normal, Jump, Powerhit 중 하나인가?), (U가 입력되었는가?), (y좌표가 244인가?=땅에 닿아있는가?)의 3가지 조건을 만족하면 점프를 실행한다. 점프는 플레이어의 y방향 속도를 -16으로 초기화하고, 플레이어 상태를 Jump로 바꾼다. 만약 플레이어가 공중에 떠있다면, 플레이어 y좌표에 속도를 더한 뒤, 중력을 적용하여 y방향 속도 값을 1 감소시킨다. (따라서 중력가속도는 $1px/frame$) 만약 플레이어가 $y > 244$, 즉 바닥 아래 위치한다면 $y = 244, v_y = 0$으로 초기화를 하여 바닥에 서있도록 한다. 그리고 플레이어 상태가 Dive라면 Lie로 전환하고, Dive가 아니라면 Normal로 전환한다.

3단계는 스파이크 및 다이브 연산이다. H가 입력되었다면 실행되며, 상태에 따라 나누어진다. Jump 상태일 경우, 스파이크를 실행하는데, 5프레임 동안 상태를 Powerhit로 바꾼다. (공의 스파이크는 다음 장에서 다룬다.) Normal 상태이고 L 또는 R 입력이 있을 경우, 다이브를 실행한다. 상태를 Dive로 바꾸고, 현재 입력 방향으로 다이브 방향을 설정한 뒤, y방향 속도를 -5로 바꾼다.

4단계는 상태 전환이다. Powerhit 상태가 된지 5프레임이 지났다면 Jump 상태로 전환한다. 이외에도 애니메이션 관련 연산이 있지만, 이동과는 관련 없으니 생략하겠다.

위와 같이 5단계를 거치면 플레이어 이동 연산이 종료된다. 아래 문제에서 개념을 확인하겠다.

문제 2.5.1. 플레이어의 가능한 y좌표 값을 모두 구하시오.

Proof. 플레이어의 기본 y좌표는 244이고, 바닥에서 가능한 y방향 속도는 $-16, -5$이다. 중력은 1이므로 각각 계산해보자.

y방향 속도가 -16일 때, 가능한 y좌표는 Figure 2.8와 같다. 최고점에 도달하면 대칭적으로 돌아가게

된다. 식으로 적어보면 다음과 같다.

$$\{y\} = \left\{244 - \frac{33k - k^2}{2} \middle| 0 \le k \le 16\right\} = \left\{108 + \frac{l(l+1)}{2} \middle| 0 \le l \le 16\right\}$$

y방향 속도가 -5일 때도 같은 방법으로 계산하면 다음과 같다.

$$\{y\} = \left\{244 - \frac{11k - k^2}{2} \middle| 0 \le k \le 5\right\} = \left\{229 + \frac{l(l+1)}{2} \middle| 0 \le l \le 5\right\}$$

따라서 가능한 y좌표 값은,

$$\{y\} = \{108, 109, 111, 114, 118, 123, 129, 136, 144, 153, 163, 174, 186, 199, 213, 228, 244\}$$
$$\cup \{229, 230, 232, 235, 239, 244\}$$

□

v_y	(-16)	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
y	244	228	213	190	186	174	163	153	144	136	129	123	118	114	111	109	108

v_y	(-5)	-4	-3	-2	-1	0
y	244	239	235	232	230	229

Figure 2.8 시간에 따른 y방향 속도 및 y좌표

문제 2.5.2. 1회 다이브 당 x좌표 이동량을 구하시오. (벽에는 충돌하지 않는 것으로 한다.)

Proof. 다이브의 종료는 y좌표로 판단하므로 다시 내려올 때까지의 시간을 계산해야 한다. 위 문제에서 확인했듯이 다이브 시 y좌표는 244에서 229까지 감소한 뒤, 다시 244로 돌아온다. 따라서 $5+1+5 = 11$ 프레임 후에 y좌표가 다시 244가 된다. Lie 상태로 돌아가기 위해서는 y좌표가 244보다 커져야 하므로 1 프레임을 더하면 총 12 프레임 동안 이동하게 된다. 따라서 x좌표 이동 거리는 $12 \times 8 = \mathbf{96}$px이다. 하지만, 다이브를 실행하려면 반드시 H와 함께 LR을 실행해야 하므로, 상태가 Dive로 바뀌기 전 1 프레임의 LR까지 입력된다. 따라서 Dive 상태가 아닌 1프레임의 입력까지 더하면 전체적으로 $96+6 =$ **102**px만큼 이동하게 된다. □

문제 2.5.3. 플레이어의 가능한 x좌표 값을 모두 구하시오.

Proof. 우선, 1p와 2p의 시작 x좌표는 $36, 396$으로 필드를 기준으로 대칭이다. 또한, 코트도 32 184, 248 400으로 필드를 기준으로 대칭이다. 따라서 1p의 가능한 x좌표 값만 구하고 대칭하면 된다.

1p 기준으로 보자. x좌표를 바꿀 수 있는 커맨드는 LR과 다이브 뿐이다. LR은 x좌표를 6만큼 바꾸고, 다이브는 x좌표를 102만큼 바꾼다. 102는 6의 배수이므로, 다이브는 고려하지 않아도 된다. 따라서 우선 플레이어는 시작 x좌표와 6의 배수만큼 차이나는 다음 집합으로 이동할 수 있다.

$$\{36 + 6k | k \in \mathbb{Z}, 32 \le 36 + 6k \le 184\}$$

위 집합은 저 구간에서 모든 6의 배수의 집합으로, 최솟값은 36, 최댓값은 180이다. 이제 벽과 충돌해서 x좌표가 32와 184로 바뀌는 경우를 생각하자. 32와 184는 6으로 나눈 나머지가 2와 4이므로 플레이어는 32와 184 사이의 6으로 나눈 나머지가 2와 4인 모든 x좌표로 이동할 수 있다. 따라서 다음과 같이 32와 184 사이의 6으로 나눈 나머지가 0, 2, 4인 x좌표들, 즉 모든 짝수들이 답이 된다.

$$\{2k | k \in \mathbb{Z}, 32 \leq 2k \leq 184\}$$

이제 다음과 같은 사실을 얻는다. 우선, 시작할 때는 6으로 나눈 나머지가 0이지만 왼쪽이나 오른쪽 벽과 충돌 후에는 2 또는 4가 된다. 따라서 벽 충돌 전에 이동했던 x 좌표로는 벽 충돌 후에 다시는 이동할 수 없다. 6으로 나눈 나머지가 2 또는 4인 좌표로는 왼쪽이나 오른쪽 벽에 충돌하여 언제든지 돌아갈 수 있다. □

이것으로 플레이어 이동 알고리즘이 끝났다. 이제 입력만 주어지면 시간에 따른 플레이어의 위치를 계산할 수 있을 것이다.

2.6 기초 게임 표기법

이 절에서는 게임의 표기법을 소개한다. 필자가 개발한 **기초 게임 표기법**은 하나의 피카츄 배구 게임을 표기하는 방식으로 다음과 같은 규칙을 가진다.

1. 1p와 2p의 커맨드는 슬래쉬(/)로 구분한다. (왼쪽이 1p의 커맨드이다.)
2. 프레임 별 입력은 대쉬(-)로 구분한다.
3. 동일한 커맨드의 반복은 커맨드 마지막에 반복할 숫자를 붙이는 것으로 표현한다.

예를 들어, 2p 썬더의 경우 /L-/23-/L30-/ULH-/14-/DH10로 표현된다. 이 기초 게임 표기법은 **피카츄 배구 서브 프로그램**에서 직접 실행할 수 있다. 이 기초 게임 표기법은 서브를 개발하거나 게임을 시뮬레이션할 때 유용하게 사용된다. 다음 장부터 나올 문제들은 이런 커맨드와 기초 게임 표기법을 바탕으로 하니, 이 프로그램을 사용하여 이론을 확인해보는 것이 좋을 것이다.

3장: 공의 운동

공은 피카츄 배구에서 가장 중요한 물체라고 할 수 있다. 공격과 수비, 승리와 패배 모두 공에 의해 일어난다. 모든 기술도 공을 매개로 하므로 공의 움직임을 이해하면 피카츄 배구를 이해한 것이다. 4장부터는 3장의 내용을 반드시 필요로 하므로 3장은 완벽하게 숙지하도록 한다.

3.1 공

공이란, 피카츄 배구의 핵심 오브젝트인 배구공을 말한다. 배구공은 포켓몬스터 테마에 맞게 포켓볼 모양으로 생겼다. 스프라이트의 크기는 40×40이고, 기본 스프라이트 5개와 추가 스프라이트 2개로 구성된다. 기본 스프라이트 5개는 다양한 회전 상태의 공이고, 추가 스프라이트는 스파이크 후 하이퍼볼 상태일 경우 공의 궤적을 표현한다. 여기서 공의 회전은 불연속적임을 알 수 있다. 또한, 하이버볼 상태의 공이 회전 상태 배열에 들어가 있어서 스파이크를 하지 않았는데도 공이 가끔 투명해지는 버그가 존재한다. 공도 플레이어와 마찬가지로 항상 공의 중심 좌표를 사용한다.

공은 게임이 시작될 때 위치와 속도가 초기화된다. 전 판을 1p가 이겼을 경우 $(56, 0)$ 위치로, 2p가 이겼을 경우 $(432 - 56, 0) = (376, 0)$위치로 초기화된다. 속도의 경우 $(0, 1)$로, 즉, 떨어지는 방향으로 시작한다. 공은 위치와 속도가 모두 2차원 벡터로 표현된다.

공도 플레이어와 마찬가지로 상태를 가지는데, 기본 상태(Normal)와 하이퍼볼(Hyperball)로 2개이다. 플레이어가 공을 스파이크할 경우, 하이퍼볼 상태로 전환된다. 자세한 건 뒤에서 다룬다.

공의 경우, 커맨드가 존재하지 않고 플레이어의 움직임에 따라서 움직이는 오브젝트이다. 따라서 바로 공의 위치를 결정하는 알고리즘을 다루자. 공의 위치를 결정하는 알고리즘은 공의 이동 알고리즘과 플레이어-공 충돌 알고리즘으로 나뉜다.



Figure 3.9 공 스프라이트

3.2 공의 이동

공의 이동은 3단계로 구성된다. 1단계는 벽 충돌 판정이다. 먼저 x좌표의 경우, 예상 x좌표(x좌표+x속도)가 20보다 작거나 432보다 크다면 x좌표 속도를 반전한다. y좌표의 경우, 예상 y좌표가 0보다 작다면 y방향 속도를 1로 초기화한다. 여기서 2가지 사실을 알 수 있다. 첫째, x좌표는 탄성 충돌이지만 y좌표는 비탄성 충돌, 즉 속도를 잃게 된다. 둘째, x좌표 충돌 범위가 비대칭적이다. 공의 반지름이 20px이므로 공은 왼쪽 벽을 통과하지 못하지만, 오른쪽 벽은 통과할 수 있다. 이로 인해서 1p보다 2p 수비가 더 어려워진다.

2단계는 네트 충돌 판정이다. 이전 x좌표가 191와 241 사이이고, y좌표가 176보다 크다면 네트와 충돌한다. 즉, 실질적 네트는 $192 \leq x \leq 240, y \geq 177$에 존재한다고 할 수 있다. 보이는 네트는 $213 \leq x \leq 220, 176 \leq y \leq 279$까지이므로(1장 참조), 실질적 네트가 더 크다. 또한, 실질적 네트는 $x = 216$ 기준으로 대칭이므로 네트 판정은 보이는 네트와 달리 대칭이다. 네트 충돌 판정은 2가지로 나뉜다. $177 \leq y \leq 192$인 경우, 네트의 꼭지와 충돌한 판정이 된다. 이 경우에는 y방향 속도가 반전된다. 위의 경우가 아니라면, 네트 기둥과 충돌한 판정이 된다. 이 경우, x좌표가 216보다 작다면 x좌표 속도를 음수로 두고, 216 이상이라면 양수로 둔다.

3단계는 x, y좌표 계산이다. 우선 y좌표에 y방향 속도를 더한다. 이때, y좌표가 252보다 커진다면, 바닥에 충돌한 판정으로, 이후 게임이 종료된다. (주의!!!) 이 경우, x좌표 연산은 생략하고 y좌표는 252로 초기화되며, y방향 속도는 반전된다. 게임이 끝나기 직전 공의 궤도를 보면 갑자기 수직으로 꺾는 것을 관찰할 수 있다. 아니라면, x, y좌표에 속도를 더하여 좌표를 계산한 뒤, y방향 속도에 1(중력가속도)를 더한다.

공이 바닥에 닿았다면, 그 라운드가 종료된다. 공의 x좌표가 216보다 작으면 2p의 승리, 216 이상이면 1p의 승리이다. 따라서 **정확히 정중앙에 떨어지면 1p의 득점**이다.

3.3 플레이어와 공의 충돌

플레이어와 공의 충돌은 이 게임의 하이라이트라고 보아도 좋다. 이 절에서 스파이크의 정확한 메커니즘을 알아볼 것이다. 플레이어와 공의 충돌이 발생할 조건은 2가지이다. 첫번째로, 플레이어와 공의 x, y좌표 차이가 모두 32 이하이어야 한다. 이는 공의 중심이 플레이어의 테두리에 도달할 때이다. (플레이어의 크기가 64×64) 두번째로, 이번 프레임에서 공과 플레이어 간 충돌이 일어나지 않았어야 한다. 이는 공이 피카츄 내부로 들어갈 경우 버그가 일어날 것을 방지하는 조건이며, 속공을 연습했다면 알고 있었을 것이다. 또한, 플레이어와 공 충돌 연산에서는 공의 속도값만이 바뀌며 충돌 연산은 1p부터 일어나는 것에 유의하자.

플레이어와 공의 충돌은 3단계로 구성된다. 1단계는 x방향 속도 계산으로, 다음 공식에 따라 계산

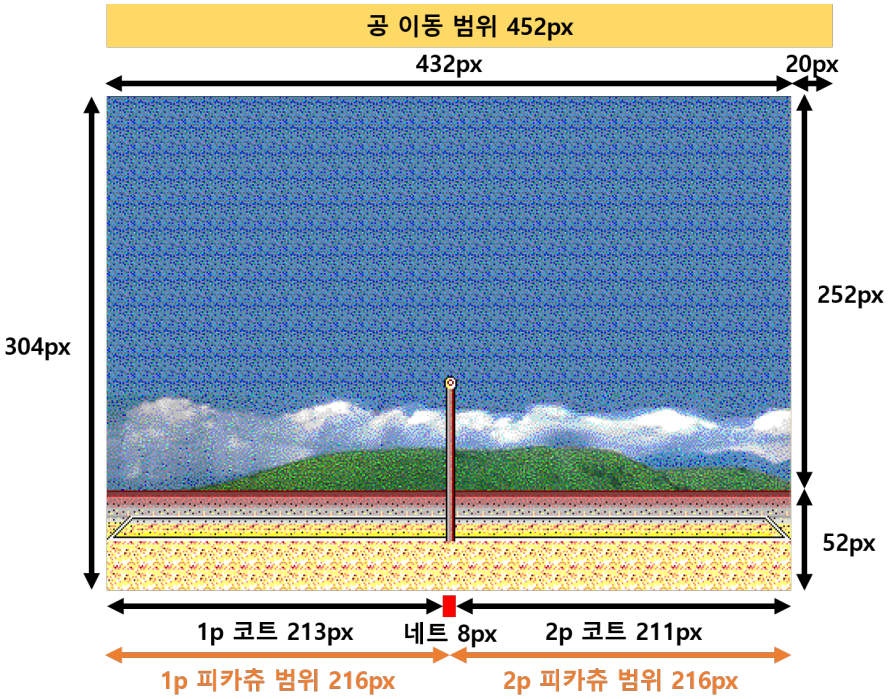


Figure 3.10 공과 플레이어의 이동 범위

된다.

$$v_x = \begin{cases} +\left\lfloor \frac{|x_{공} - x_{플레이어}|}{3} \right\rfloor & \text{if } x_{공} > x_{플레이어} \\ -\left\lfloor \frac{|x_{공} - x_{플레이어}|}{3} \right\rfloor & \text{if } x_{공} < x_{플레이어} \\ v_x & \text{if } x_{공} = x_{플레이어} \end{cases} \tag{3.1}$$

우선, 속도의 크기(절댓값)은 공과 플레이어의 x좌표 차이를 3으로 나눈 몫으로 결정된다. 그리고, 방향(부호)은 공이 더 왼쪽에 있으면 왼쪽으로, 오른쪽에 있으면 오른쪽으로 결정된다. 이때, 이렇게 결정된 v_x 값이 0이라면, v_x를 **-1, 0, 1 중 랜덤한 값**으로 바꾸게 된다. 이는 대전 중 공이 좌우로 움직이지 않는 상태로 경기가 끝나지 않는 상황을 방지하기 위함이다. 예외적으로, 공과 플레이어의 x좌표가 같다면 x방향 속도는 보존된다. 이는 땅굴기와 속공에서 자주 나타난다.

2단계는 y방향 속도 계산이다. 다음과 같은 공식을 사용한다. (아래의 등호는 대입의 의미이다.)

$$v_y = -\max(|v_y|, 15)$$

단순히, 기존 y방향 속도를 음수로 만들되, 크기가 15보다 작았으면 -15로 만드는 과정이다. 따라서 **공과 피카츄가 (스파이크 없이) 충돌할 경우 공은 반드시 위로 올라간다.** 또한, 기존 y방향 속도가 크다면 이를 유지하므로, 넉넉한 범위에서 위치 에너지가 보존된다고 할 수 있다.

3단계는 대망의 스파이크이다. 플레이어가 Powerhit 상태일 때 이 연산을 진행한다. 공은 우선 Hyperball 상태로 전환된다. (만약 Powerhit 상태가 아니라면, 공은 Normal 상태로 전환된다.) I_x

는 x방향 커맨드 값으로, L일 경우 -1, R일 경우 1, 커맨드가 없을 경우 0의 값을 가진다. I_y는 y 방향 커맨드 값으로, U일 경우 -1, D일 경우 1, 없을 경우 0의 값을 가진다. (커맨드가 중복될 경우 $I_x(LR) = -1, I_y(UD) = -1$로 정의한다.) 이때, x방향 속도는 다음과 같은 공식을 사용한다.

$$v_x = \begin{cases} +(|I_x| + 1) \times 10 & \text{if } x_{공} < 216 \\ -(|I_x| + 1) \times 10 & \text{if } x_{공} \geq 216 \end{cases}$$

공이 1p코트 또는 2p코트에 있는지에 따라서 방향이 결정되고, 크기는 좌우 커맨드 입력이 있을 경우 20, 없을 경우 10으로 결정된다. 이때 주의할 경우는 $x = 216$인 상황인데, 뒤의 예제에서 자세히 다루겠다.

y방향 속도는 다음 공식으로 계산된다.

$$v_y = |v_y| \times I_y \times 2$$

커맨드 방향으로, 원래 속력의 2배 크기만큼 **가속**된다. 이 공식에서 유일한 가속이 일어난다. 원래 y 방향 속도가 빠를수록 새로운 속도도 더 빨라지므로, 다음과 같은 사실을 얻는다. **공의 y 방향 속도가 가장 빠른 지점, 즉 공이 가장 아래로 내려왔을 때 스파이크를 해야 Hyperball의 속도가 더 빨라진다.**

3.4 정리

알고리즘 순서에 주의하자. 한 프레임에서 우리가 배운 3가지 알고리즘은 **공 이동**→**플레이어 이동**→**플레이어와 공의 충돌**의 순서로 진행된다.

이제 피카츄배구 물리학의 기본 개념은 완전히 끝났다. 알고리즘을 모두 이해했다면, 커맨드만 주어졌을 때 게임 상황 전체를 계산하여 재현하는 것이 가능할 것이다. 앞으로의 절에서는 이 물리법칙으로 발생하는 특별한 상황들을 다룰 것이다.

3.5 연습문제

문제 3.5.1. 공이 $(216, 0)$ 좌표에 있고 속도는 $(0, 1)$이라 한다. 플레이어의 간섭이 없을 때 공의 시간에 따른 위치를 구하시오.

Proof. $x = 216$일 경우, 네트와의 충돌이 일어나므로 이에 주의하자. 네트와 충돌하기 전까지는 등가속도 운동을 한다. 따라서 프레임 t일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, t + 1), (x, y) = (216, 1 + 2 + \cdots + t) = (216, \frac{t(t+1)}{2})$$

네트는 $y \geq 177$부터 시작되므로, 다음을 얻는다.

$$\frac{t(t+1)}{2} \geq 177 \Rightarrow t \geq 19$$

따라서 $t = 19$일 때 $(216, 190)$에 도달하고, $t = 20$일 때 네트와의 충돌이 일어난다. $177 \leq y \leq 192$이므로 네트 꼭지와 충돌한 판정이 되며, y방향 속도가 반전된다. 따라서 $t = 20$일 때 좌표는 $(216, 190 - 20) = (216, 170)$이고, 속도는 $(0, -20 + 1) = (0, -19)$가 된다. 이제 다시 등가속도 운동으로 올라간다. 프레임 $20 + t$일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, -19+t), (x, y) = \left(216, 190 + (-20) + (-19) + \cdots + (-20 + t) = 170 - 20t + \frac{t(t+1)}{2}\right)$$

이번에는 천장과 충돌할 때를 본다.

$$170 - 20t + \frac{t(t+1)}{2} < 0 \Rightarrow t \geq 14$$

프레임 33일 때, 위치는 $(216, 1)$, 속도는 $(0, -6)$이다. 따라서 프레임 34일 때 예상 y좌표가 음수이므로 위치는 $(216, 2)$, 속도는 $(0, 2)$가 된다. 이제부터 다시 등가속도 운동을 한다. 프레임 $33+t$일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, t + 1), (x, y) = \left(216, 1 + 1 + 2 + ... + t = 1 + \frac{t(t+1)}{2}\right)$$

잘 살펴보면 이전의 공식에서 y좌표가 1씩 늘어났다. 같은 과정을 반복하면, 프레임 $52(t = 19)$일 때 $(216, 191)$에 도달하고, 꼭지와 충돌하여 y방향 속도가 반전된다. 이후도 정확히 같으므로, 프레임 $53 + t$일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, -19 + t), (x, y) = \left(216, 171 - 20t + \frac{t(t+1)}{2}\right)$$

두번째 운동은 단순히 첫번째 운동의 y좌표에 1을 더한 것이다. 세번째 운동도 계산해보면 그렇다. 프레임 $66 + t$에서 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, t + 1), (x, y) = \left(216, 2 + \frac{t(t+1)}{2}\right)$$

다시 $t = 19$일 때를 보면, $(216, 192)$에 도달하여 충돌이 발생하고 반복된다. 프레임 $86+t$일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, -19 + t), (x, y) = \left(216, 172 - 20t + \frac{t(t+1)}{2}\right)$$

프레임 $99 + t$일 때, 속도와 좌표는 아래와 같다.

$$(v_x, v_y) = (0, t + 1), (x, y) = \left(216, 3 + \frac{t(t+1)}{2}\right)$$

$t = 19$일 때를 보면, $(216, 193)$에 도달한다. 이제는! 네트 꼭지 판정이 아니라 네트 기둥 판정이다. 네트 기둥 판정은 x방향 속도를 바꾸는데, 0이므로 아무것도 바뀌지 않는다. 따라서 네트를 뚫은 것이나 다름 없다. 공은 이대로 바닥으로 향한다. $t = 21$일 때, $(216, 234)$에 도달하고 $t = 22$일 때(프레임 121) $234 + 22 = 256 > 252$이므로, 위치는 강제로 $(216, 252)$로 이동하고, $x \geq 216$이므로 1p의 승리로 게임은 끝난다.

게임은 끝났지만 이후도 계산해보면, 정확히 $(216, 0)$에 도달하고, 처음의 상태로 돌아가는 것을 확인할 수 있다. 주기가 142인 주기 수열이다. □

문제 3.5.2. 다음 커맨드를 실행했을 때, 게임이 종료되는 프레임을 구하시오. (단, 1p 서브이고 2p 플레이어는 없는 것으로 한다.)

R/-/20-R/24-U/6-RH/

Proof. 모든 알고리즘을 필요로 하는 문제이다. 차근차근 살펴보자.

1p는 $(36, 244)$에서 시작한다. R을 실행할 경우, $(42, 244)$로 이동하고 21 프레임까지 이 자리에 정지한다. 공은 $(56, 0)$에서 시작하여 자유 낙하할 것이다. 프레임 t에서 공의 위치는 $(56, \frac{t(t+1)}{2})$이다. 피카츄와 공이 충돌하려면 x, y좌표 차이가 모두 32 이하여야 하므로, 공의 y좌표가 212이상이 되는 순간을 보면 된다.

$$\frac{t(t+1)}{2} \geq 212 \Rightarrow t = 21, (x, y) = (56, 231), v_y = 22$$

따라서 21프레임일 때 충돌이 일어난다. 플레이어는 Powerhit 상태가 아니므로, 단순 충돌이 일어날 것이다. 공식에 대입하면 다음과 같다.

$$v_x = \left\lfloor \frac{56 - 42}{3} \right\rfloor = 4, v_y = -22$$

22프레임부터는 플레이어가 오른쪽으로 이동한다. 1프레임 당 6만큼 이동하므로 45프레임(24프레임 후)에는 $42 + 6 \times 24 = 42 + 144 = 186 > 184$이니 $(184, 244)$까지 이동한다. (네트에 막힘) $21 + t$ 프레임에서는 공의 y좌표가 아래와 같다.

$$v_y(21 + t) = -22 + t, y(21 + t) = 231 + (-22) + \cdots + (-23 + t) = 231 - 23t + \frac{t(t+1)}{2}$$

예상 y좌표가 0보다 작아지는 순간은 $t = 16$일 때이며, $t = 15$일 때 좌표는 다음과 같다.

$$(x, y) = (56 + 4 \times 15, 231 - 23 \times 15 + 120) = (116, 6)$$

따라서 $t = 16$, 37프레임에는 $(120, 7)$에 도달한다. $36 + t$프레임에서 공의 좌표는 아래와 같다.

$$(x, y) = (116 + 4t, 6 + \frac{t(t+1)}{2}), v_y = t + 1$$

45프레임($t = 9$)에서는 공이 $(152, 51)$에 위치한다. 이제부터 플레이어는 점프를 시작하는데, 45+t에서 플레이어의 위치는 아래와 같다.

$$(x, y) = (184, 244 - 17t + \frac{t(t+1)}{2})$$

51프레임에서 공의 좌표는 $(176, 126)$, 플레이어의 좌표는 $(184, 163)$으로, 아직 충돌이 일어나지 않는다. 52프레임에서 공의 좌표는 $(180, 142)$이고 플레이어의 좌표는 $(184, 153)$으로 충돌이 일어난다.

이때, 플레이어의 커맨드는 RH이므로 스파이크가 실행된다. 공식에 대입하면 공의 속도는 다음과 같다.

$$v_x = 2 \times 10 = 20, v_y = 17 \times 0 \times 2 = 0$$

따라서 53프레임에서 공의 좌표는 $(200, 142)$이고, 53+t프레임에서 공의 좌표는 아래와 같다.

$$(x, y) = (200 + 20t, 142 + \frac{t(t+1)}{2})$$

t에 값을 대입해보면 네트와의 충돌은 없음을 확인할 수 있다. 이제 공은 2p코트로 넘어간다. $t = 12$일 때, 예상 x좌표가 $440 > 432$이므로 x방향 속도가 반전되어 $(420 - 20, 142 + 78) = (400, 220)$에 도달한다. 따라서 $64 + t$프레임에서 공의 좌표는 아래와 같다.

$$(x, y) = (420 - 20t, 142 + \frac{(t+11)(t+12)}{2})$$

$t = 4$일 때, 예상 좌표는 $(340, 262)$로 공이 바닥에 충돌하며, 68프레임에 게임은 종료되고 이 때 공의 좌표는 $(360, 252)$가 된다.

□