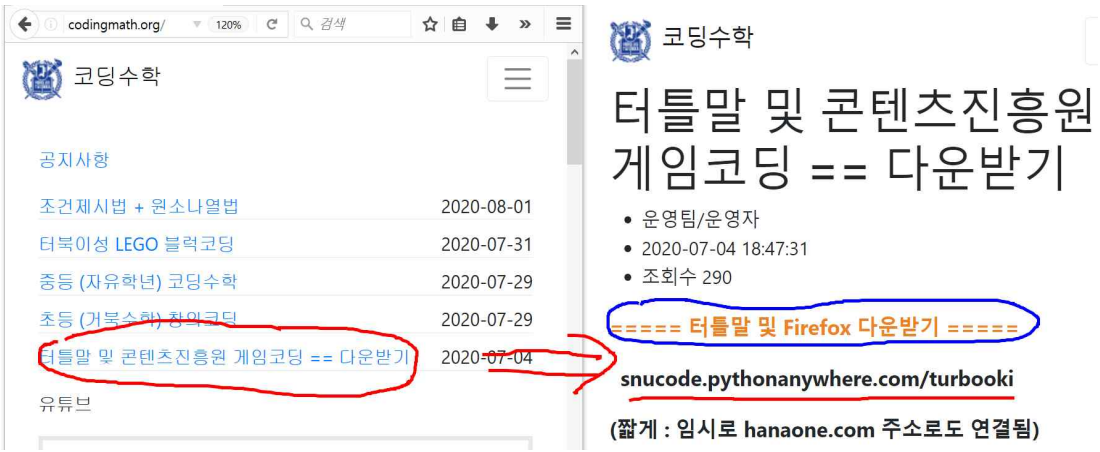
터틀크래프트 + 터틀말 = 터북이 코딩수학

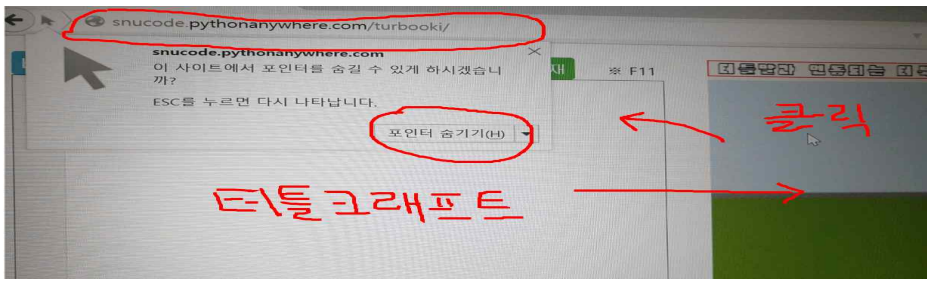
● 홈페이지 : codingmath.org snucode.org hanaone.com

Firefox52down.zip 다운받고 Firefox 사용함



● codingmath.org 주소에서, Firefox52down.zip 다운받고 푼다

위와 같은 codingmath.org 화면의 다운받기 게시판에 들어간 후, Firefox52down.zip 링크를 사용하여 zip 파일을 다운받고, zip을 풀면 나타나는 파이어폭스시작.vbs 를 클릭하여 http://snucode.pythonanywhere.com/turbooki/ 주소로 갑니다 (임시로 hanaone.com 도 가능). 이때 코딩 편집기가 나타나는데, 만일 [beginxyz] 단추를 누르면 오른편에 마인크래프트와 같은 세상이 나타납니다. 마우스로 나타난 오른편 화면을 클릭하면 아래와 같은 [포인터 숨기기] 알림창이 나오는데, 마우스로 포인터 숨기기를 클릭하면 3차원 터틀크래프트 세상으로 들어가게 됩니다.

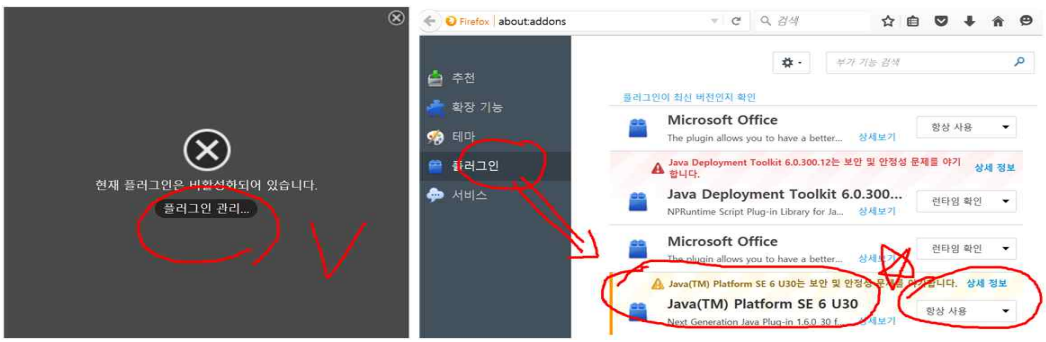


쉬프트키 shift 를 누르면 중력이 작동하여 (마인크래프트의 스티브에 해당하는) 터북이가 땅으로 내려오게 됩니다. 마인크래프트에서와 같이 WASD 키를 사용할 수 있으며, Z키로 올라가고, C키로 내려오고, X키로 뛰고, 블록을 만들고 깨면서 플레이를 할 수 있습니다.

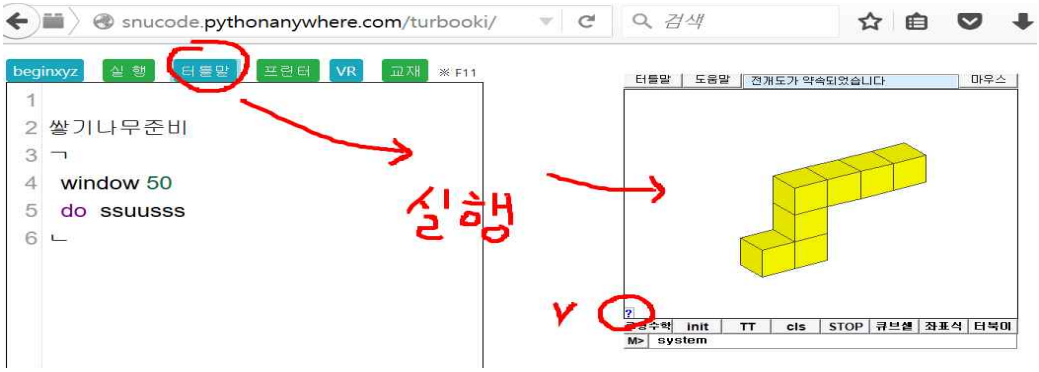
Shift키, Space키, W, S, A, D키, Z, X, C키, Ctrl+마우스, Alt+마우스

파이어폭스로 [터틀말] 거북이를 불러내고, [터틀말] 창의코딩을 하자

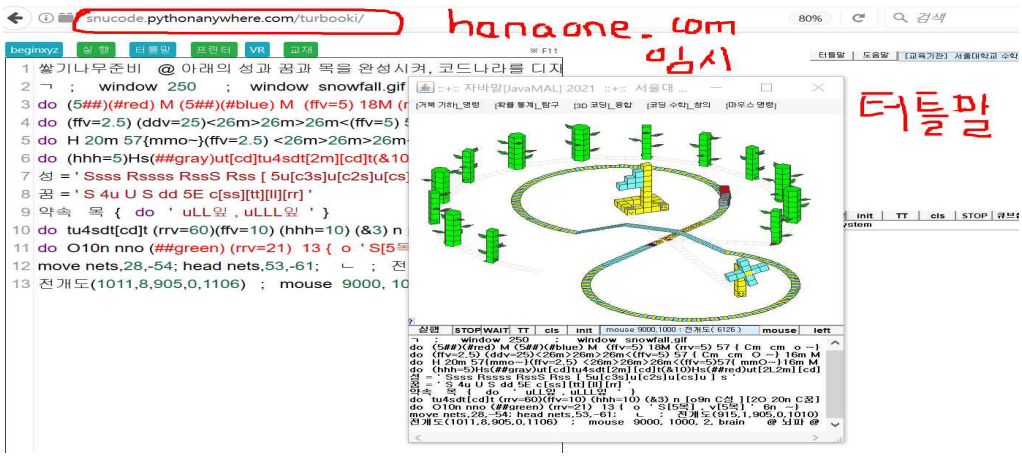
이제 [터틀말] 이라고 쓰인 단추를 누릅니다 (터틀말은 크롬에서 작동하지 않기에 우리는 Firefox52down.zip을 풀면 나오는 파이어폭스를 사용합니다). 만일 아래 화면과 같이 [플러그인 관리] 알림창이 나오는 경우, [플러그인 관리] 를 클릭한 후에 나타나는 아래 오른쪽과 같은 화면에서 Java SE 6U30 을 [항상 사용] 으로 선택하고 다시 시작합니다.



이제 아래와 같이 쌓기나무준비 아래에 쓰여진 거북명령을 [터틀말] 단추로 실행시키면 아래와 같이 오른쪽에 나타나는 터틀말 화면에서 쌓기나무 그림이 그려집니다.

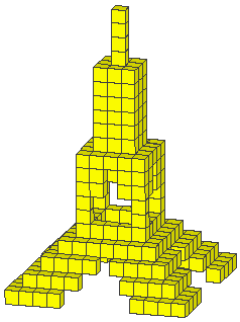


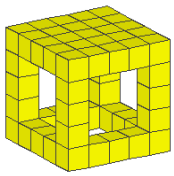
터틀말 화면 아래의 [?] 표시를 마우스로 클릭하면, 큰 터틀말 화면이 나온다. 그 곳 메뉴에서 창의코딩 컨테스트 코드를 실행하면 철도를 따라 기차가 움직이는 모습을 볼 수 있다.

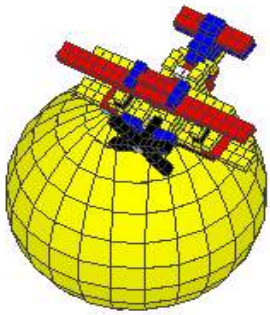


터북이 창의코딩 : 창의융합 코딩컨테스트 갤러리

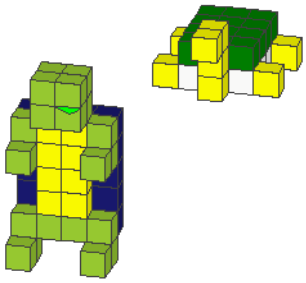
에펠탑



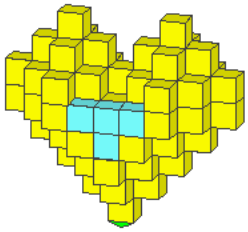




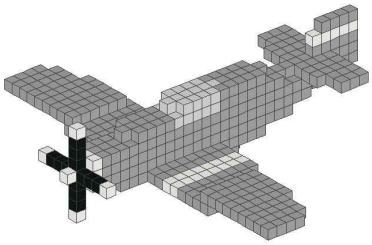
비행기

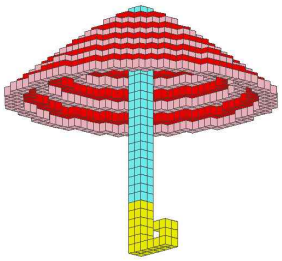


터틀말 거북

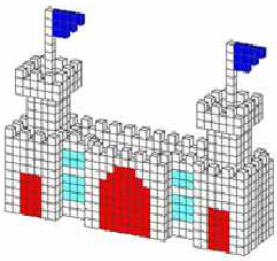


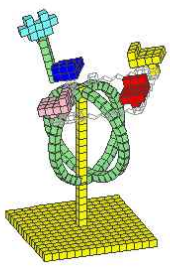
하트

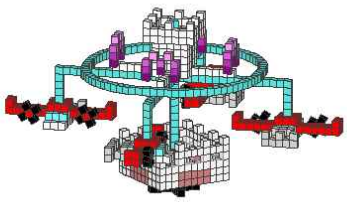




비행기



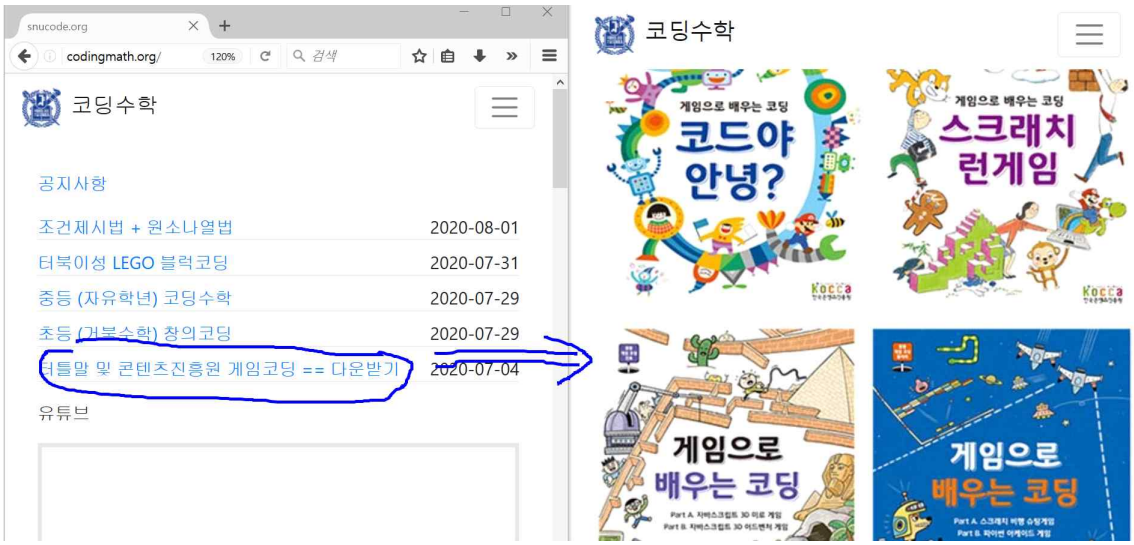




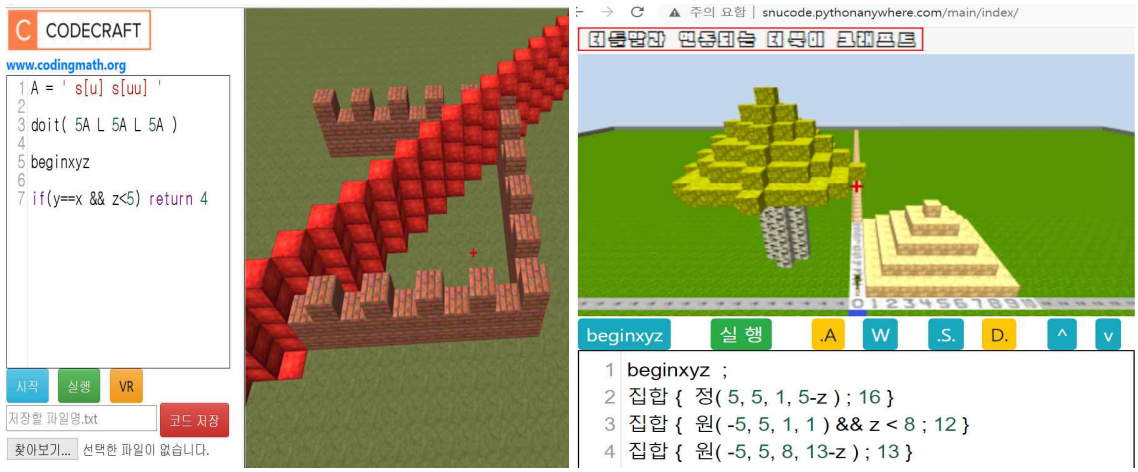
성 놀이공원

터틀크래프트 : 창의융합 터북이 코딩수학

(1) codingmath.org 주소에 가서 터틀말 및 콘텐츠진흥원 게임코딩 다운받기를 클릭하면 오른편에 터틀말과 문화체육관광부 산하 한국 콘텐츠진흥원을 통해 공개한 코드크래프트 코드와 게임코딩 교재를 다운받는 링크가 나온다. 참고로 2018년에 공개한 코드크래프트를 수학학습과 터틀말과 맞도록 새로 만든 것이 터틀크래프트 (TurtleCRAFT) 이다. codingmath.org 주소에도 터틀크래프트를 쓸 수 있는 링크와 게시판 설명 (동영상)이 제공되고 있다.

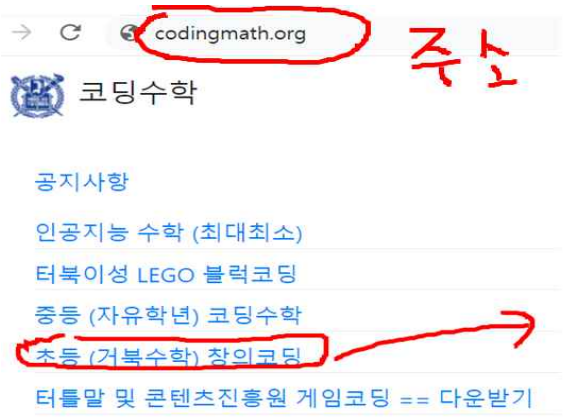


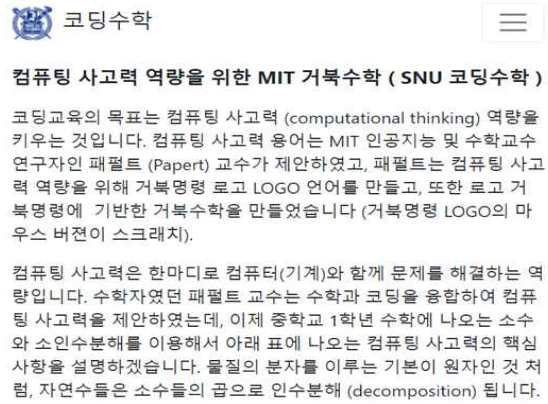
터틀크래프트에서는 수학학습과 코딩학습을 융합하는 정직구원 입체 집합명령을 기반으로 입체를 만들지만, 콘텐츠진흥원을 통해 공개한 코드크래프트는 수학적 수식을 자바스크립트 코딩에 담아서 입체를 만든다. 다음은 코드크래프트와 터틀크래프트 화면의 모습이다.

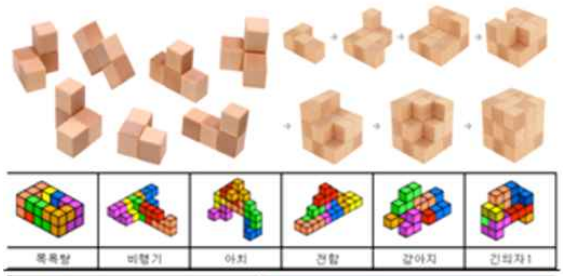


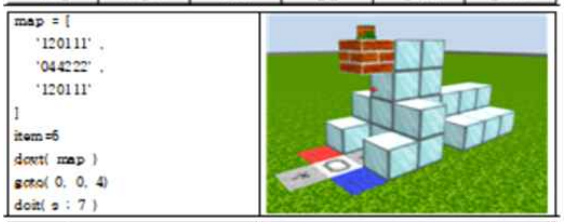
((참고)) 위 오른쪽에 있는 정직구원 입체 집합명령에 기반한 코딩은 초등학생도 사용할 수 있도록 독자적으로 코딩수학 명령체계를 연구하여 개발한 것이다. 반면 위 왼쪽의 코딩 표현인 if(y==x && z<5) return 4 표현은 자바스크립트의 수식 표현을 쓴 것이다 !!

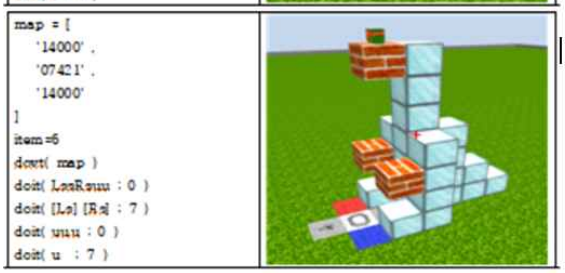
코딩수학 (codingmath.org) 게시판 교재로 학습한다

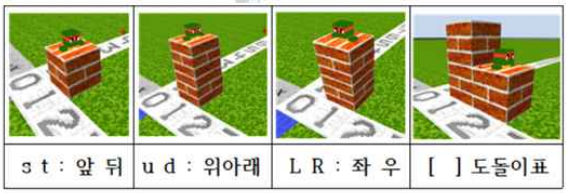


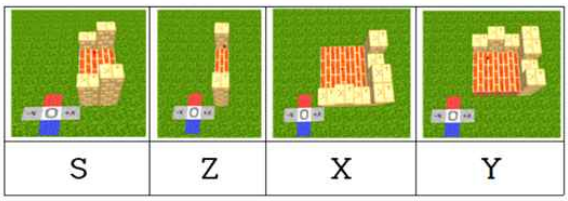


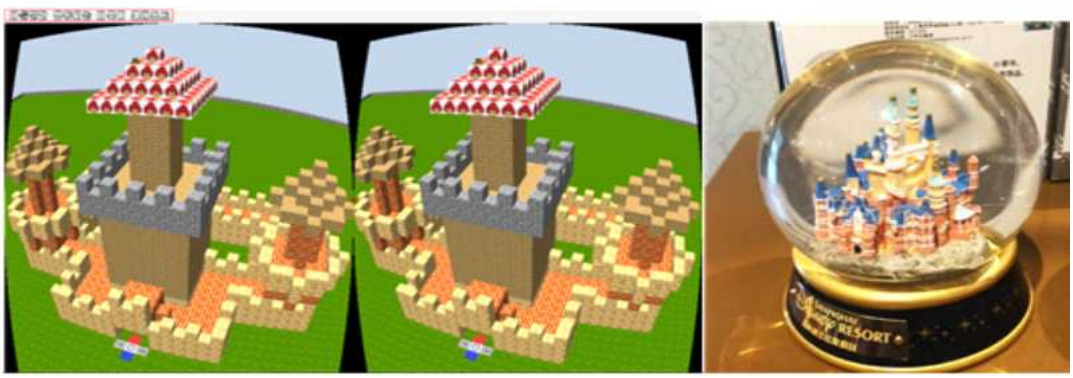






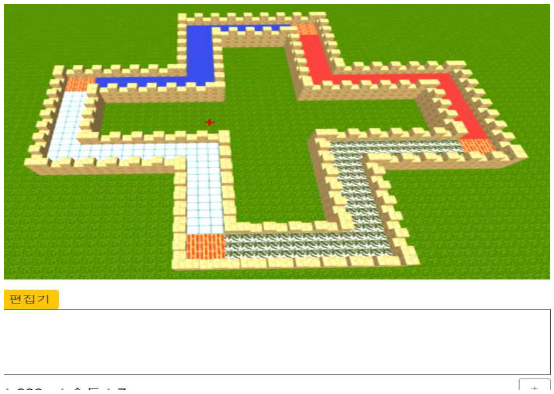






스크래치 코드와 사각형 모양 성을 만드는 아래 터틀크래프트 코드를 비교하세요.

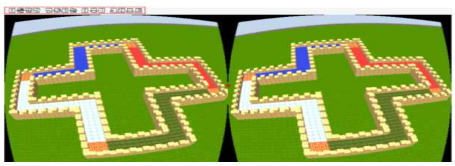
위에서는 대문자 A에 s[u]s[uu] 명령을 저장시킨 후, 문자 A로 성벽을 그렸다. 반복 4 { d 거북명령 } 문법은 { 와 } 사이에 있는 거북명령을 4번 반복하라는 것이다. 스크래치의 repeat 반복명령과 같다. 아래는 doitsxyz 명령 후, 대문자 S, X, Y, Z 로 성 모양을 만드는 터틀크래프트 코드이다. 아래에 사용된 while(n<20) { 거북명령 } 은 { 와 } 사이의 거북 명령을 반복시키는 프로그래밍 언어 문법이다. A='s[u]s[uu]' 치환문자와 n=n+1 과 같은 표현은 코딩과 수학을 연결하는 창의융합 코딩수학이다.



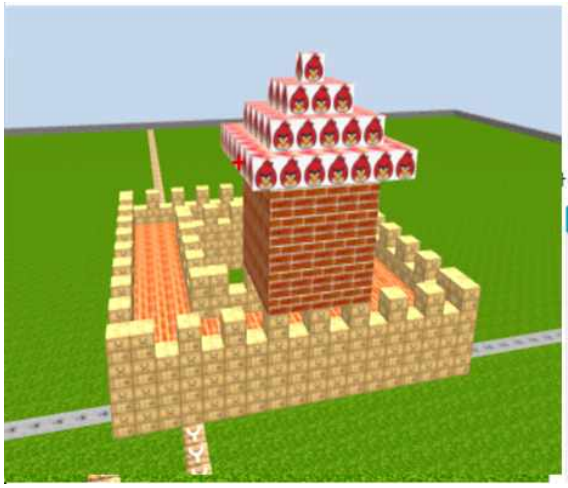
편집기

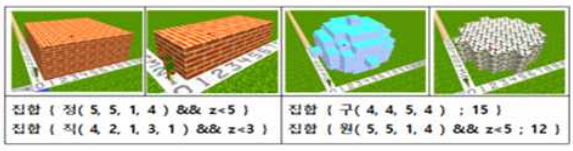
beginxyz 실행 편집기

```
doitsxyz   // 거북에게 SXYZ 사용을 알림

n = 3        // 변수 n 의 값이 바닥의 모양

반복 4 {
     doit( 5SX 5SY 5S : n )   // 성의 4분의 1
     doit( X )                // 왼쪽으로 90도 회전
     n = n+1        // n 값을 1 만큼 증가시킴
 }
```



위와 같이 만든 성 모양에 아래와 같은 성탑을 만들고, 나무도 심고, 또한 VR 놀이도 하자. 성탑을 만들 때, beginxyz 아래에 집합 명령을 사용하면 편하다. 아래와 같은 모양을 거북 명령으로 하나하나 만들면 너무 힘들다 (이 것을 원소나열법 코딩이라 부른다). 아래와 같이 정직구원 집합 명령을 사용하여 고등학교 수학에 나오는 조건제시법으로 코딩을 해보자.





래의 [beginxyz] 단추를 누르세요. 이어서 [실행] 단추도 누르세요.

beginxyz 실행 편집기

```
doitsxyz   // 거북에게 SXYZ 사용을 알림
반복 4 {    doit( 5S X )    }

beginxyz  :
집합 { 정( 6, 1, 5, 2 )  &&  z<10 }
// 맨 아래의 중심이 (6,1,5) 이고
// 10 층 아래로 반지름이 2인 탑
집합 { 정( 6, 1, 10, 13-z ) ;  18 }
// 10층 부터 시작해서, 각각의
// z 층에 반지름 13-z 로 만듦
```

집합 { 원(−5, 5, 8, 13−z) ; 13 } 명령은 맨 아래의 중심의 좌표가 (−5,5,8) 이며, 여기로부터 시작해서 각각의 z 층에 중심이 (−5,5) 이고 반지름이 13−z 인 원을 만든다.

ginxyz 실행 편집기

```
doit( LLsuuuu [ss]u[ss] ; 18 )  ; beginxyz  ;
집합 {  정( 5, 5, 1, 5-z ) ; 16 }
집합 {  원( -5, 5, 1, 1 ) && z < 8 ; 12 }
집합 {  원( -5, 5, 8, 13-z ) ; 13 }
```

코딩수학

beginxyz 실행 편집기

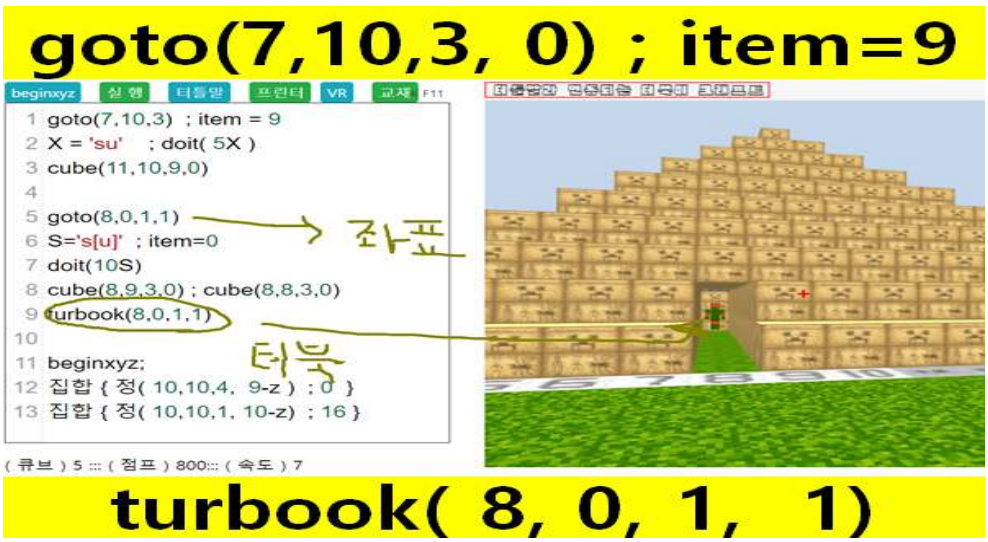
```
doit( LLsuuuu [ss]u[ss] ; 18 )
// doit  거북 원소나열법 (실행 단추로)
beginxyz  ;   //  조건제시법 (시작 단추로)
if(z<0 || z>15) return 0      // 속도 개선
집합 {  정( 5, 5, 1, 5-z ) ; 16 }
// 1층 이상이며, 16번 고대 벽돌로
// 각 층에 중심(5,5), 반지름 5-z
집합 {  원( -5, 5, 1, 1 ) && z < 8 ; 12 }
// 1층 이상이면서 && 8층 미만의 층까지
집합 {  원( -5, 5, 8, 13-z ) ; 13 }
```

집합 { 정(5, 5, 1, 5-z) ; 16 } 명령은 맨 아래의 중심점 좌표가 (5,5,1) 이며, 여기서 부터 시작해서 각각의 z 층에서 중심이 (5,5) 고 반지름 (가로와 세로쪽 반지름)이 5-z 인 **정**사각형을 만들라는

거북이와 터북이를 움직이는 goto 와 turbook 명령

doit 명령을 실행하는 **닌자 거북이 (turtle)** 와 마인크래프트 스티브와 같이 세상을 바라보는 **터북이 (turbook)** 를 goto 와 turbook 명령으로 이동을 시키자. 이 때, 몸을 좌우로 회전시키고 또한 위 아래로 회전시키는 것도 가능하다. 아래는 먼저 집합 명령으로 속이 비어 있는 피라미드를 만든 후, goto 명령으로 특정 위치에 가서 doit 명령으로 만드는 모습이다.

다음, 아래는 만든 피라미드에 item=0 하고 doit 명령과 cube 명령으로 구멍을 내는 코드이다. 여기에 turbook(8,0,1,1) 과 같은 명령으로 터북이를 특정 위치로 보내는 모습이 있다. goto(1,2,3) 명령은 닌자 거북이를 (1,2,3) 에 보내되, 몸의 방향을 0직각 즉 0도 방향인 동쪽 방향으로 잡아준다. turbook(1,2,3, 1.5) 명령은 터북이를 (1,2,3) 에 보내되, 몸의 방향을 1.5직각 즉 135도 방향으로 잡아준다 (**거북이 방향은 n직각 즉 n x 90 도로 계산**).

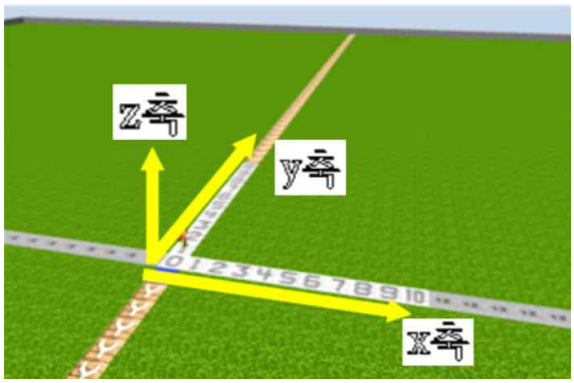


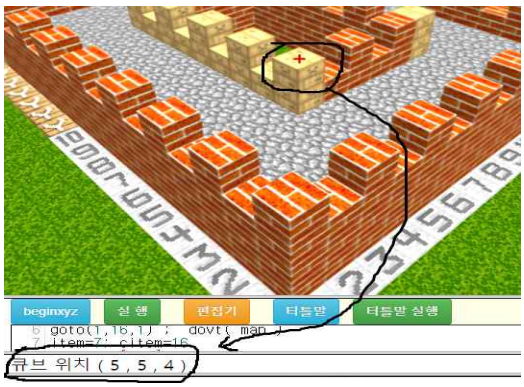
((탐구)) turbook(8,0,1, 1, −2) 하면 터북이는 물구나무를 서서 세상을 보게 된다. 닌자 거북이는 항상 서있지만, 터북이는 45도 회전도 하고 위 아래로 물구나무도 할 수 있다.

== (좌표) : 터틀 크래프트와 수학 교과서에서의 3차원 좌표 표현 ==

좌표 알아내기 : 마우스 클릭을 하면 십자가 + 표시의 위치에 해당하는 3차원 좌표가 표시됨

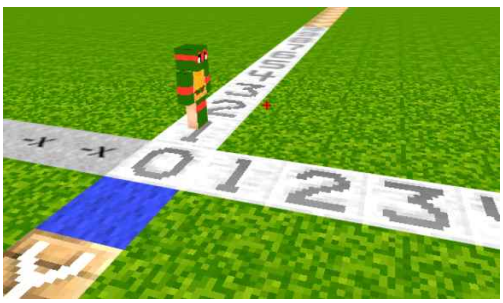
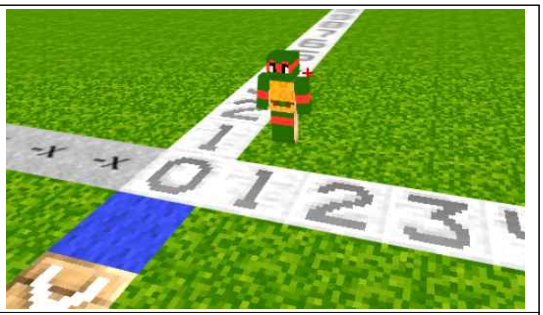
beginxyz ; 와 같이 beginxyz 다음에 ; 를 찍고 [beginxyz] 단추를 누르면 x와 y좌표축이 나타난다. beginxyz 만 쓰고 아래에 명령이 없으면 지하세계가 있는 게임 환경이 나온다 !!!



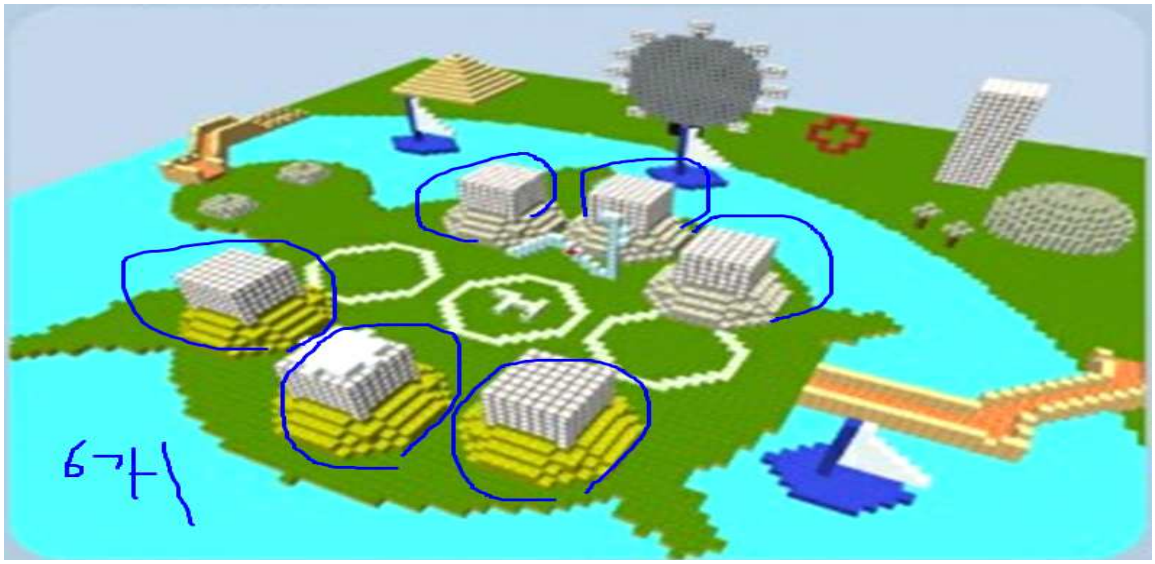


((3차원 좌표 시스템)) : (터틀크래프트) 에서는 고교 수학 교과서의 방법과 같이 (1) 동쪽 방향에 x축 (2) 북쪽 방향에 y축, 그리고 (3) 하늘 방향에 z축을 대응시켜 좌표를 표현한다. goto(1,2,3) 명령을 하면 좌표 (1,2,3) 으로 거북이가 된 후에, 자동으로 0도 방향 (양의 x축, 동쪽) 으로 향한다. 만일 goto(1,2,3,2) 하면, 거북의 방향을 2 직각 180도 방향으로 맞춘다.

goto 명령과 head 명령 (거북이를 특정 좌표로 보내고, 원하는 방향을 향하도록 하는 명령)

	
goto(0, 1, 1) **자동으로 마지막에 head(0) 실행**	**goto(1, 1, 1) ; head(3)** **간단하게 goto(1, 1, 1, 3) 으로 씀**
거북이가 좌표 (0,1,1) 로 간 후, 자동으로 0도 즉 동쪽 방향을 향함.	거북이가 좌표 (1,1,1) 로 간 후, 3 직각 (270도) 방향으로 향함.

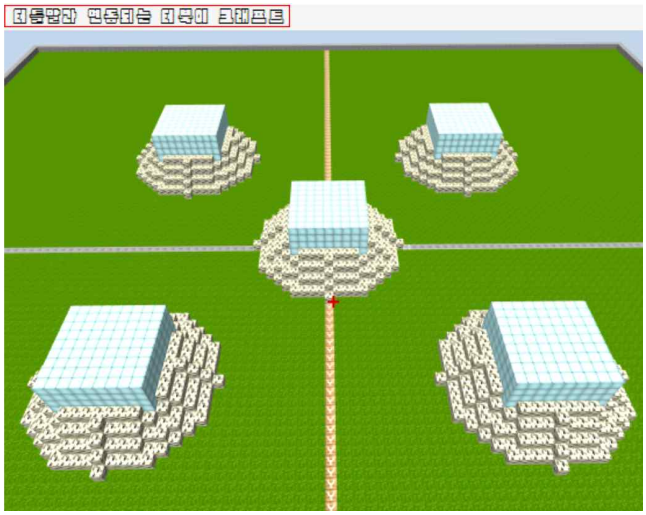
[십자가 + 표시와 @ 사용] 화면의 **십자가 + 표시** 장소로 닌자 거북이를 보내려면, 먼저 그 위치에서 마우스를 클릭하여 그 곳의 좌표를 알림창에서 얻는다 (바로 위에서 클릭해야 정확한 값이 나옴). 그 곳의 좌표가 (0,0,1) 이라면 편집기 맨 처음에 goto(0,0,1) @ 과 같이 쓰고 [실행] 단추를 누르면, 기호 @ 있는 곳 까지만 실행이 되어, 좌표 (0,0,1) 위치로 닌자 거북이가 이동한다. 이와 같이, **[실행] 단추로 간단하게 코드를 테스트할 경우에 @ 기호를 쓰면 편하다.**



((탐구)) 위는 시흥 초등학생의 창의코딩 작품이다. 시흥 앞바다에 거북이 섬을 만들고, 그 위에 서울대 시흥캠퍼스를 만들고, 다리로 연결하고, 해안가에 보트와 허니문카 놀이동산 등의 시설을 만들었다. 거북이 섬에 있는 6개의 건물의 모양이 비슷하다. 이 것을 학교 수학의 변수와 함수를 사용하여 코딩수학으로 만들어보자.

beginxyz | 실 행 | 터틀말 | 프린터 | VR | 교재 ※F11

```
1  beginxyz ;
2  if(z<0) return 0
3
4  함수 건물(a,b,c) {
5    집합 { 정(a,b,4, 4) && z<8 ; 6 }
6    집합 { 원(a,b,c, 10-z) && z<5 ; 12 }
7  }
8
9  k=건물(20,20,1) ; if(k>0) return k
10 k=건물(20,-20,1) ; if(k>0) return k
11 k=건물(0,0,1) ; if(k>0) return k
12 k=건물(-20,20,1) ; if(k>0) return k
13 k=건물(-20,-20,1) ; if(k>0) return k
```

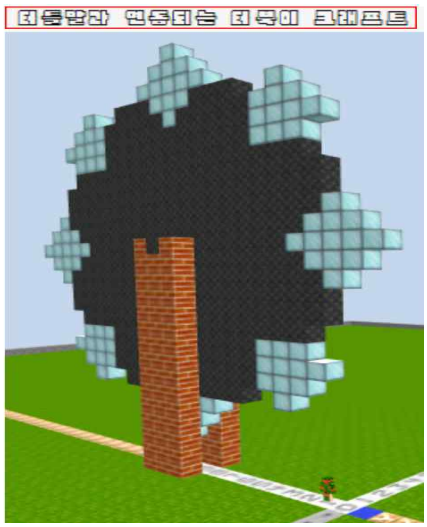


((설명)) 2줄의 if(z<0) return 0 는 z 좌표의 값이 0보다 작은 지하 세계를 모두 0 으로 (즉 비어있는 공간으로) 만들라는 것이다. 비어있게 만드는 것이 실행속도를 빠르게 하기에, 실행속도 개선을 위해 beginxyz 아래에 이런 명령을 사용한다. 4줄부터 7줄까지 좌표 (a,b,c) 에 대한 건물 함수가 있는데, 주어진 좌표를 중심으로 4층부터 6번 블록으로 건물을 만들고, 4층 아래에는 원(a,b,c, 10-z) 명령으로 조그만 동산을 만든다. 건물 함수는 자판기와 같이 주어진 좌표 (a,b,c) 에 대해 어떤 값을 내주는데, 그것을 k=건물(20,20,1) 과 같이 k 로 받는다. k 가 0 보다 크면 그 곳에 return k 값으로 작은 동산과 건물을 만든다.

((탐구)) 앞에 제시된 시흥 초등학생의 창의코딩 작품에서, 해안가에 있는 허니문카 놀이동산 시설을 어떻게 만들 수 있을까? 해안선과 거북섬을 만리장성과 같은 성벽으로 연결시키는 방법은 무엇인가?

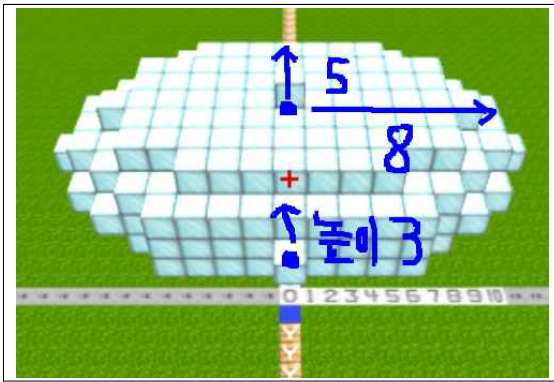
beginxyz 실 행 터틀말 프린터 VR 교재 ※ F11

```
1  beginxyz ;
2  if(z<0 || z>30) return 0
3
4  n=0
5  반복 8 {
6   집합 { 구(0,10+10*cos(n,1),
7    15+10*sin(n,1), 1,2,2)  && x==0 ; 6 }
8   n=n+45
9  }
10
11 집합 { 구(0,10,15,  1,10,10)  ; 46 }
12 집합 { 직(-1,10,1,   0,1,   15) }
13 집합 { 직(1,10,1,    0,1,   15) }
```

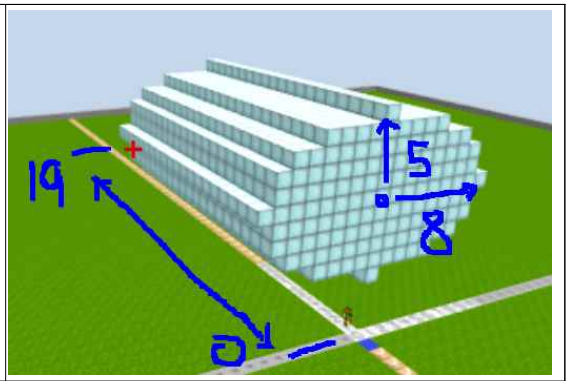


((허니문카 코드 설명)) 먼저 실행속도 개선을 위해 z<0 인 지하세계와 z>30 으로 30층보다 높은 곳을 0 으로 비워둔다 (실행속도 개선이 중요). 허니문카를 45도 간격으로 배치한다. 허니문카를 먼저 만들고 나중에 뼈대를 만드는 이유는? beginxyz 아래의 명령에서는 먼저 만들어지는 것이 우선적으로 보인다. 즉 허니문카 뼈대를 먼저 만들고 나중에 허니문카를 만든다면 먼저 만들어진 뼈대에 가려져서 8개의 허니문카가 잘 안보일 수 있다.

고등학교 수학에 나오는 sin cos 함수를 사용하여 n=0; 반복 8 { ... ... ; n=n+45 } 반복 명령으로 0도, 45도, 90도 등등 45도 간격으로 8개 허니문카의 중심 좌표를 cos(n,1) cos(n,1) 등으로 각각 잡는다. 그냥 cos(n) 하면 고등학교 수학의 radian 라디안 값에 의해 sin cos 가 계산된다. 45도 각도에 대한 sin cos 값은 sin(45,1) 등으로 계산한다. 허니문카의 원판 모양의 뼈대는 중심 좌표가 (a,b,c) 인 경우에 구(a,b,c, x축 방향의 반지름, y축 방향의 반지름, z축 방향의 반지름) 명령으로 구면체 (동그란 구를 좌우 상하로 늘리고 줄여서 만드는 입체) 모양으로 만든다. 다음은 다양한 경우의 구 집합명령에 대한 예이다. 특별히 구(0,0,10, 8, 5) && y<20 명령은 y 좌표가 0부터 시작해서 20 미만까지 즉 19까지 (반지름이 각각 8과 5인) 누워있는 (타)원기둥을 만든다.

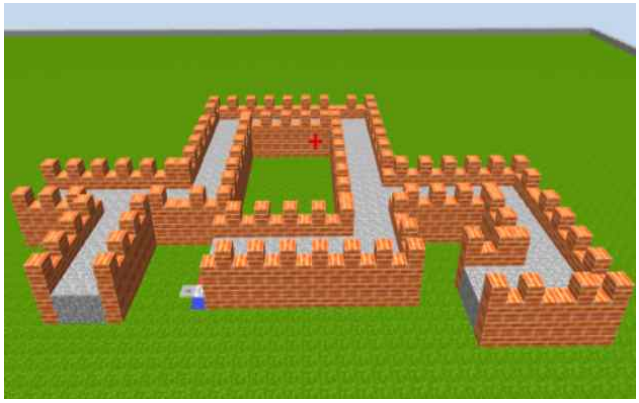


집합 { 구(0,0,10, 8, 5, 3) ; 6 }

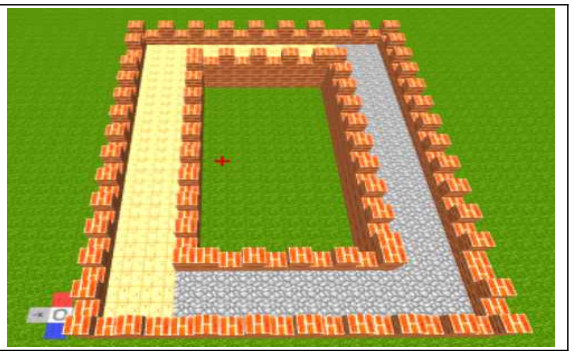


집합 { 구(0,0,10, 8, 5) && y < 20 ; 6 }

((**거북섬과 해안선을 연결하는 만리장성 성벽 만들기 : beginxyz 위의 doitsxyz 명령**))

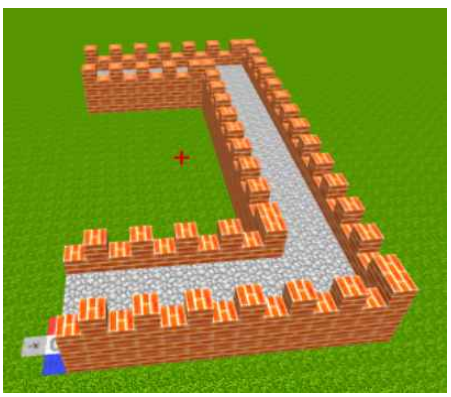


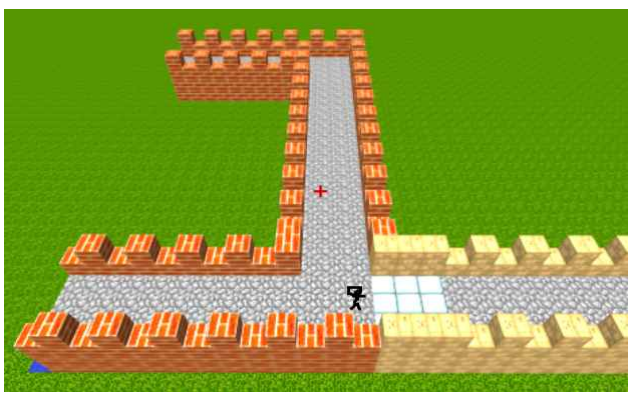
[성벽 코딩 : 치환문자로 미로와 성벽을 한꺼번에 만들기] 초중등 컴퓨팅 사고력 역량을 위한 창의코딩을 위해, 성벽 미로의 모양을 치환문자로 추상화하고, 그 것으로 성벽 미로를 만들자.
왼쪽과 같은 모양의 성벽 미로를 만들 때, **doitsxyz 명령을 먼저 한 후**, 똑바로 2칸을 가면서 좌우에 성벽을 만드는 것을 S로, 왼쪽으로 90도 방향을 바꾸는 미로를 X로, 오른쪽으로 90도 방향을 바꾸는 미로를 각각 Y로 만들 수 있다. 이러한 **치환문자 S, X, Y, Z 는 doitsxyz 명령어로 자동으로 생성된다.** 각각의 치환문자가 나타내는 doit 명령은 참고 페이지에서 볼 수 있다.

doitsxyz item = 9 ; citem = 7 goto(5,1,1) doit(5S X 10S X) doit(5S X 10S X ; 16)	

(설명) doitsxyz 명령을 쓰면, S는 item 의 블록으로 미로의 길을 그리고 양쪽의 성벽은 citem 의 블록으로 만든다. S는 길이가 2인 길이로, X는 가로 세로가 3인 미로의 길을 왼쪽으로 90도 회전하며 만든다. Y는 반대로 오른쪽으로 90도 회전하면서 미로의 길을 만든다.

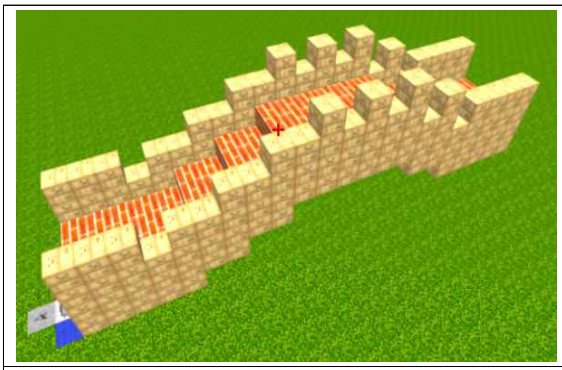
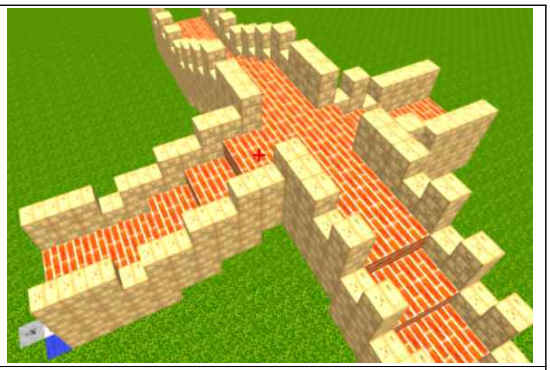
[만리장성 미로게임을 고치기] 만일 아래 왼쪽 사진과 같은 미로를 완성한 후, 나중에 아래 오른쪽과 같이 고치려면 어떻게 할까 ? 먼저 화면의 + 표시를 아래 오른쪽 사진의 거북이 표시에 놓고, 그 위치의 좌표를 알아낸다. 그 위치 1층에 goto(13,1,1) 로 가서 코딩을 시작한다



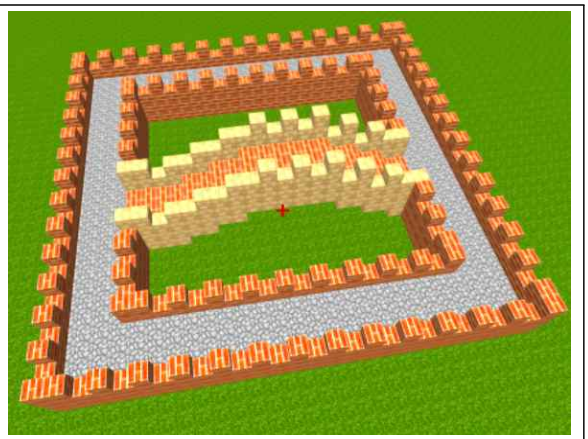


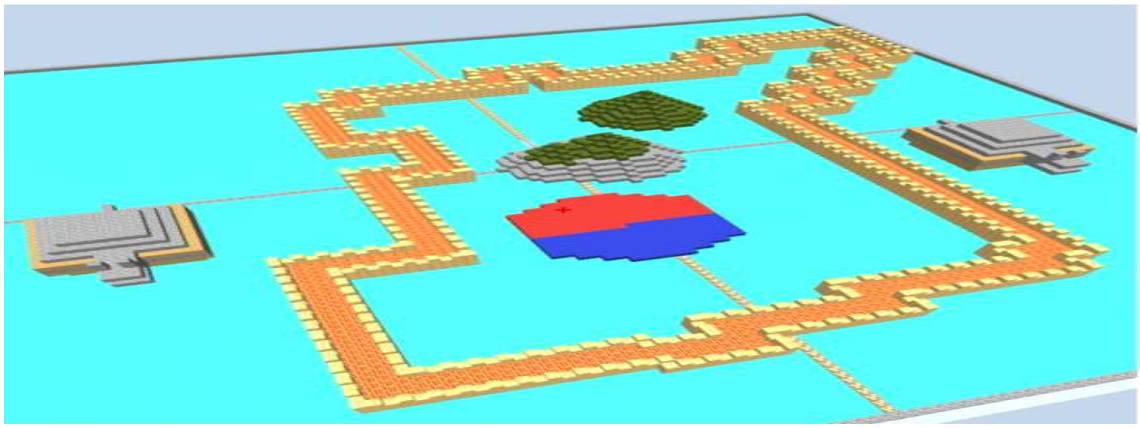
doitsxyz item = 9 ; citem = 7 goto(0,1,1) doit(5S X 10S X 5S) **goto(13,1,1)** **item = 6 ; citem =16** **doit(ZZZ)** **doit(5S ; 9)**	(1) 앞의 오른쪽 사진의 거북 표시의 좌표를 알아낸 후, goto(13,1,1) 로 닌자 거북이를 1층에 보낸다. (2) doitsxzy 의 Z는 앞에 있는 성벽을 부수며 새로운 미로 길을 만드는 문자이다. 바닥의 item=6 로 성벽을 citem=16 으로 바꾸고 대문자 ZZZ 한다. (3) 바닥을 9번으로 하는 길을 위해 doit(5S ; 9) 명령을 한다. 만일 성벽을 바꾸려면 citem 조정 (힌트) 미로의 길의 기본 모양을 먼저 만들고, 나중에 + 위치로 가서 Z로 벽을 부수고 새 길을 낸다 !

[만리장성 미로게임 창의코딩] doitsxyz 명령 후, 거북을 u, d 로 움직이며 S와 Z로 여러 가지 창의적인 미로의 길을 만들 수 있다. 예를 들어, 산을 올라가는 만리장성 미로길을 만든다.

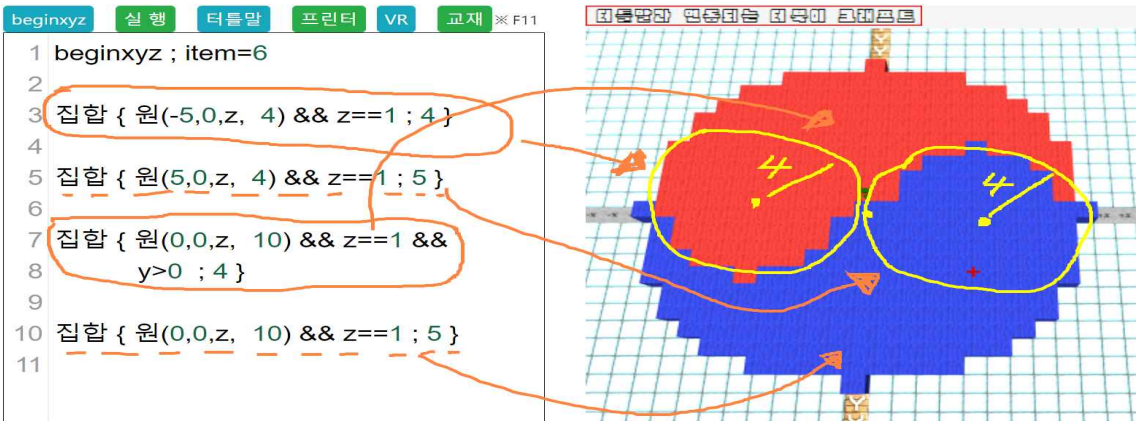
	
doitsxyz **goto(0,1,1)** **doit(ZZZ SuSuSuSSSdSdS ZZZ)** **(설명) doitsxyz 이 후, S는 길이가 2인 길, Z는 길이가 1인 길 (앞을 부수며 길을 만듦)** **u 와 d 로 거북을 움직이며 S, Z를 써보자.**	**doitsxyz ; goto(0,1,1)** **doit(ZZZ SuSuSuSSSdSdS ZZZ)** **goto(13,2,4) ; head(1)** **doit(ZZZ SdSdSd SSSS)** **goto(13,0,4) ; head(3)** **doit(ZZZ SdSdSd SSSS)**

[만리장성 미로게임 협동코딩] 각 사람이 만든 코딩 작품을 goto 좌표 명령으로 서로 연결하여 창의융합 코딩과제를 수행할 수 있다. 이 때, 각 학생은 다음과 같은 함수 (function) 를 하나씩 만들고, 그 것을 연결하여 협동작품을 만들 수 있다.

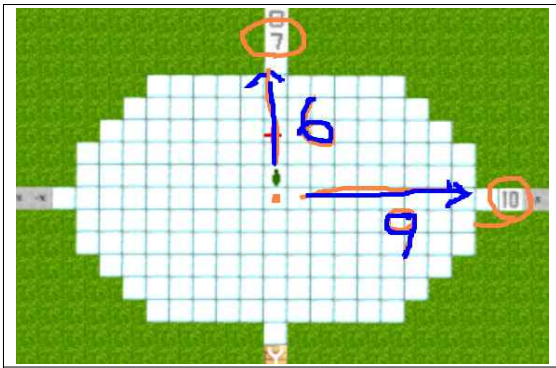
doitsxyz **function 홍길동(x,y,z) { goto(x,y,z) ;** **item=9 ; citem=7** **doit(10S X 10S X 10S X 10S X)** **}** **function 성춘향(x,y,z) { goto(x,y,z) ;** **item=7 ; citem=16** **doit(ZZ SuSuSuSSSdSdSd ZZ) }** **홍길동(0,1,1)** **성춘향(0,11,1)**	



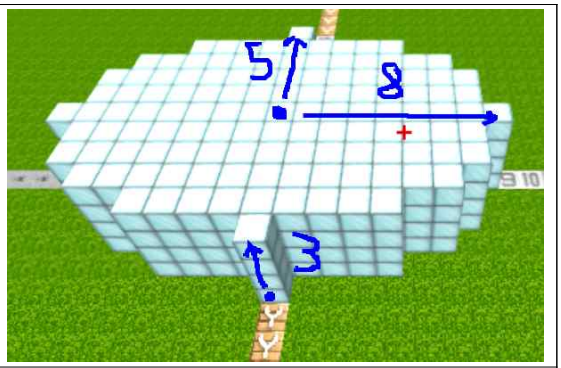
((탐구)) 위는 시흥 초등학생의 창의코딩 작품이다. 대한민국 모양을 따라 만리장성을 만들고, 가운데에 태극기와 산 그리고 바다에는 거북선이 있다. 대한민국 태극기와 척추에 해당하는 태백 산맥을 만들어보자.



((태극기 모양 만들기)) 먼저 3번 줄에 있는 집합 원 명령으로 중심좌표 (−5,0,1) 에서 반지름이 4인 빨간 원을 그린다 (블럭 4번이 빨간색이며, z==1 이라는 조건으로 1층에만 만든다). 다음으로 5번 줄의 원 명령으로 블록 번호가 5인 파란색 원을 z==1 층에 반지름이 4인 원을 만든다. beginxyz 아래의 명령에서는 먼저 만들어진 것이 끝까지 보인다. 따라서 10번 줄의 원을 먼저 만들면 전체 그림이 5번 블록으로 전부 칠해진다. 원 모양을 좌우로 늘리고 줄여서 만들 수 있는 타원을 만드는 명령은 다음과 같다.



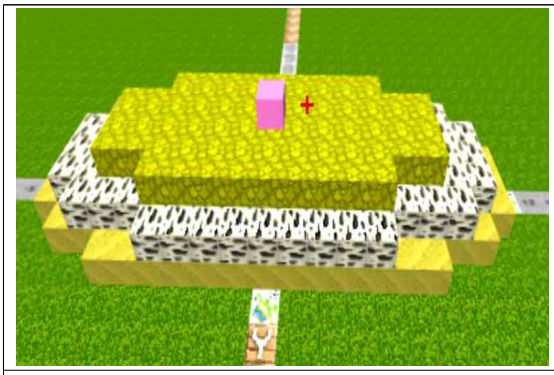
집합 { 원(0,0,0, 9, 6) && z==0 ; 6 }



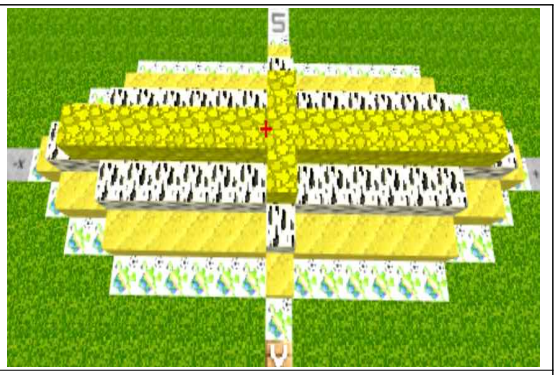
집합 { 원(0,0,1, 8, 5, 3) ; 6 }

바로 앞의 “ **집합 { 원(0,0,1, 8, 5, 3) ; 6 }** ” 명령은 z가 1층부터 1+3 층까지, 즉 z=1 이면 원(0,0,1,8,5,0), z=2 이면 원(0,0,2,8,5,0) 등등 각각의 z 층에 (타)원을 만들게 된다. “ **집합 { 원(0,0,1, 8, 5, 0) ; 6 }** ” 의 숫자 0 은 시작하는 1층에서부터 0을 더한 1+0 층까지 6번 블록으로 반지름이 각각 8과 5인 타원을 쌓으라는 것이다.

((구 명령으로 태백산맥 만들기)) 앞의 대한민국 모양에 태백산맥 척추를 만들어보자.



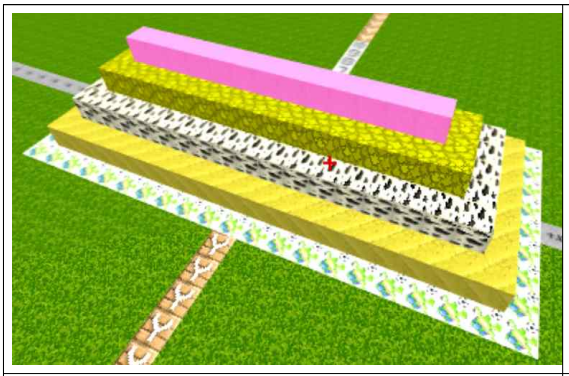
집합 { 구(0,0,0, 10, 4, 4) ; z+10 }



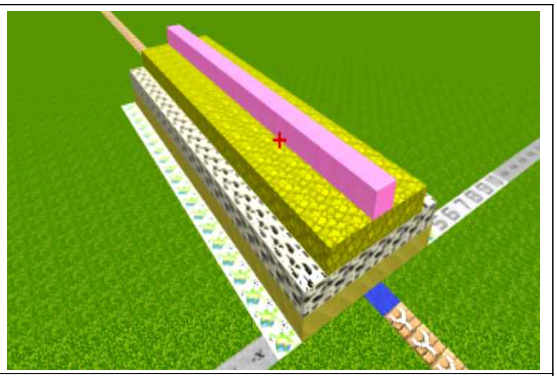
집합 { 원(0,0,0, 10-z, 4-z) ; z+10 }

바로 위의 왼쪽 명령 “ **집합 { 구(0,0,0, 10, 4, 4) ; z+10 }**” 으로 중심이 (0,0,0) 이고 x축 y축 z축 방향의 반지름이 각각 10, 4, 4 인 (구면) 입체가 만들어진다. 각각의 z 층에 z+10 번호의 블록을 쌓게하여 위와 같이 등고선 명령을 만들었다.

바로 위의 오른쪽 명령 “ **집합 { 원(0,0,0, 10-z, 4-z) ; z+10 }** ” 으로 z가 0 이상인 경우에, 즉 z=0 층에는 반지름이 각각 10−0, 4−0 인 타원, z=1 층에는 반지름이 각각 10−1, 4−1 인 타원 등등 각각의 z 층에 (타)원을 만들게 된다. z=5 층에서는 각각의 반지름이 10−5, 4−5 인 타원을 만드는데, 반지름이 4−5 와 같은 음수일 수는 없기에 z=5 층 이상에서는 (타)원을 만드는 것이 중단된다. 이러한 구와 원 명령으로 태백산맥과 비슷한 것을 만들 수 있다.



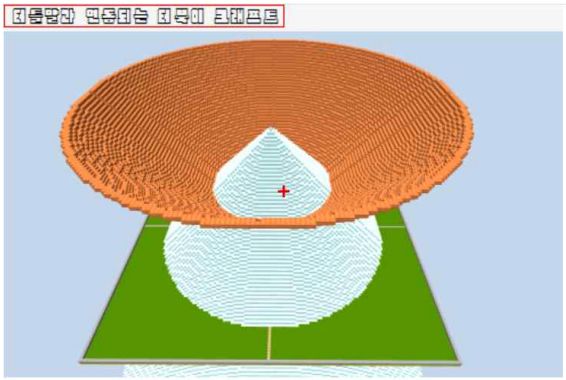
집합 { 직(0,0,0, 10-z, 4-z) ; z+10 }



집합 { 구(0,0,0, 4,4) && y<20 ; z+10 }

((참고)) 크기가 작아지는 직사각형 모양을 쌓아 올리면 피라미드와 같이 가로와 세로쪽 반지름이 각각 작아지는 사각형들이 만드는 모양을 만든다. 구(0,0,0, 4,4) 명령은 중심 좌표 (0,0,0) 이고, x축 방향의 반지름이 4이고 z축 방향의 반지름이 4인 원 모양을 만들되, y의 좌표가 0부터 y<20 까지 만든다. 따라서 오른쪽 명령에 의해 땅 밑에도 모양이 있다.

인공지능 코딩수학 (피타고라스 sqrt, abs, 집합 명령)



beginxyz 실행 편집기

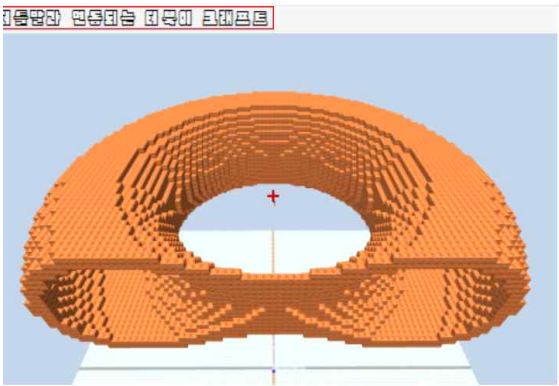
```
1 beginxyz ;
2 집합 { abs(  z , sqrt(x,y)       ) < 1  }
3 집합 { abs(  z , 50 - sqrt(x,y)  ) < 1 ;  6 }
```

코딩수학

```
2 beginxyz ;
3
4 집합 { abs(  z , sqrt(x,y)       ) < 1  }
5 // -1 < z -  sqrt(x*x+y*y) < -1 부등식 해집합
6 // sqrt(x*x+y*y) 는 x*x+y*y 의 제곱근이다.
7 // 땅에 그린 반지름 10인 원을 10층에 올림.
8 집합 { abs(  z , 50 - sqrt(x,y)  ) < 1 ;  6 }
9 // 땅에 그린 반지름 10인 원을 40층에 올림
10 // 땅에 그린 반지름 20인 원을 30층에 올림
```

위의 코드가 그린 깔대기 모양의 빨간 벽돌은 z = f(x,y) = sqrt(x,y) 라는 함수의 그래프입니다. 중학교 수학에서 y=x , y=x-3 , y=x+3 일차함수의 그래프를 그릴 때, 아래와 같이 x 와 y 의 대응표를 만든 후, 좌표 (x, y) 에 점을 찍어 그래프를 완성하게 됩니다.

z = f(x,y) = x*x+y*y 의 제곱근 = sqrt(x,y) 의 그래프를 그리려면 z와 sqrt(x,y)의 차이의 절대값인 abs(z, sqrt(x,y)) 를 사용하여 그래프의 두께를 조절하며 그린다. 집합 { abs(z, sqrt(x,y))<1 } 명령은 | z - sqrt(x,y) | < 1 인 경우의 모든 (x,y,z)를 해집합으로 모아서 그래프를 그려준다. 컴퓨터에서는 equal 등호 = 계산에 에러가 많이 생기기에 등호 y - f(x) = 0 으로 그리기가 어렵다. 대신에 | y - f(x) | < 1 과 같은 부등호로 그래프를 그린다. 컴퓨터에서는 끝없이 전개되는 무리수 등을 계산할 수 없는 한계가 있다.



beginxyz 실행 편집기

```
1 beginxyz ;  R=30  ; r=10  ; D = sqrt ( x , y )
2 집합 {  abs( (D-R)*(D-R)+z*z  , r*r ) <
3                  20 &&   y > -22     }
4 return 0
```

코딩수학

```
2 beginxyz            // 좌표식과  조건제시법
3 if(  abs(z) > 10  ) return      // 속도 개선
4 R=30  ;  r=10          // 토러스의 반지름 들
5 D = sqrt ( x , y )     //  x*x+y*y 의 제곱근
6 // 원점에서 R 만큼 떨어진 위치에, 반지름 r인
7 // 원을 만들어 나가면서 토러스 모양을 만듦
8 집합 {  abs( (D-R)*(D-R)+z*z  , r*r ) <
9                  20 &&   y > -22     }
10 return 0   // 나머지는 모두 0 (땅이 안생긴다)
```

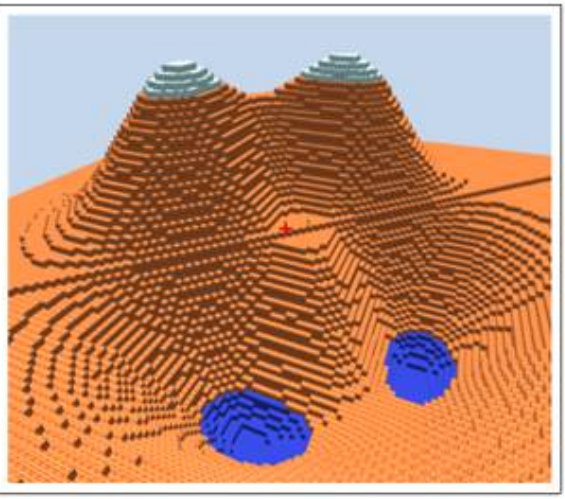
위의 코드에서 abs(z) 는 z의 절대값 |z| 을 나타내며, A와 B의 차이의 절대값은 **abs(A , B) = | A - B |** 표현으로 타나냅니다. 또한 **D = sqrt (x, y) = (x*x+y*y 의 제곱근)** 표현은 x의 제곱과 y 제곱의 합의 제곱근을 변수 D에 저장하라는 것입니다. 따라서 sqrt(x,y) 는 피타고라스 정리에 의해 원점에서 좌표 (x,y) 까지의 거리가 됩니다. 이

위의 코드는 피타고라수 정리를 사용하여 가운데가 비어있는 토러스 모양을 만든 것이다. 여기서 abs(a,b) 는 | a-b | 이다. z = f(x,y) 꼴의 2변수 함수와 그래프에서 각각의 x와 y 방향으로의 경사는 대학수학과 인공지능 수학에서 중요하다. 특히 고등학교 교과 내용이었던 부등식 영역에서의 최대 최소 문제는 인공지능 수학의 핵심적인 내용이다.

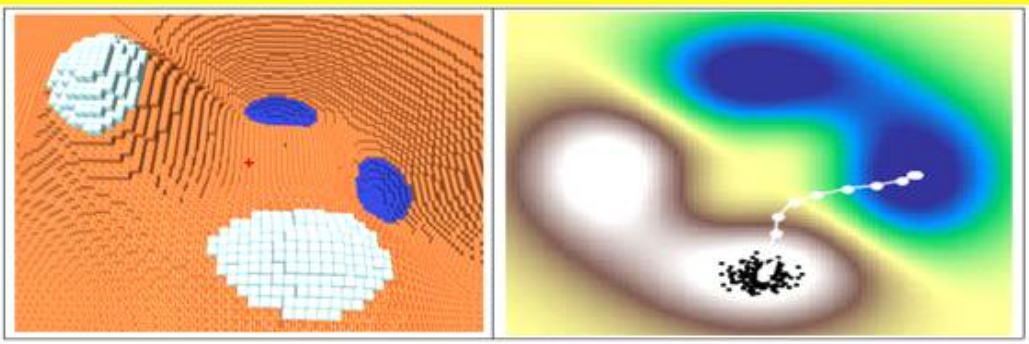
인공지능 진화전략 코딩수학

정규분포와 관계되는 $e^{-x^2-y^2}$ 와 x^3+y^3 을 곱해서 만든 2변수 함수 $f(x,y) = -3(x^3+y^3)e^{-x^2-y^2}$ 의 그래프는 아래와 같이 산봉우리와 호수가 있는 지형의 모습으로 또한 최댓값은 산 정상의 높이로 스토리텔링을 할 수 있다.

```
// print( f(1.22, 0) )
// goto(24.0,-23) // (1.22 0,-1.23)
beginxyz
window(0.05)
function f(x,y) {
 return 3*
 (-x*x*x-y*y*y)*exp(-x*x-y*y)   }
if( z<=f(x,y) && z>=f(x,y)-0.1 )  {
  if(z>1) {return 6} else if(z<-1) {
          return 5} else return 7
}
return 0
```



확률통계와 미적분 만남



위의 오른쪽 화면의 모습에서 까만 점들이 뿌려진 로봇 개미의 모습이다. 이 개미로부터 앞으로 나아갈 경사의 방향을 얻어 조금씩 나아가는 모습이 위의 그림의 경로에 나타나있다. 여기서 중요한 것이 고등학교 미분이 다루는 경사 gradient를 확률과 통계로 근사할 수 있다는 것이다. 다음은 함수 그래프의 최대 최소 문제를 다루는 진화 전략의 핵심적인 수식이다. 다음 수식을 고등학교 수준에서 최대한 스토리텔링으로 접근하는 코딩수학의 결과는 부록에서 다룬다.

$$f'(x) \approx \frac{1}{\sigma}\mathbb{E}_{\varepsilon \sim N(0,1)}\varepsilon f(x+\sigma\varepsilon)$$

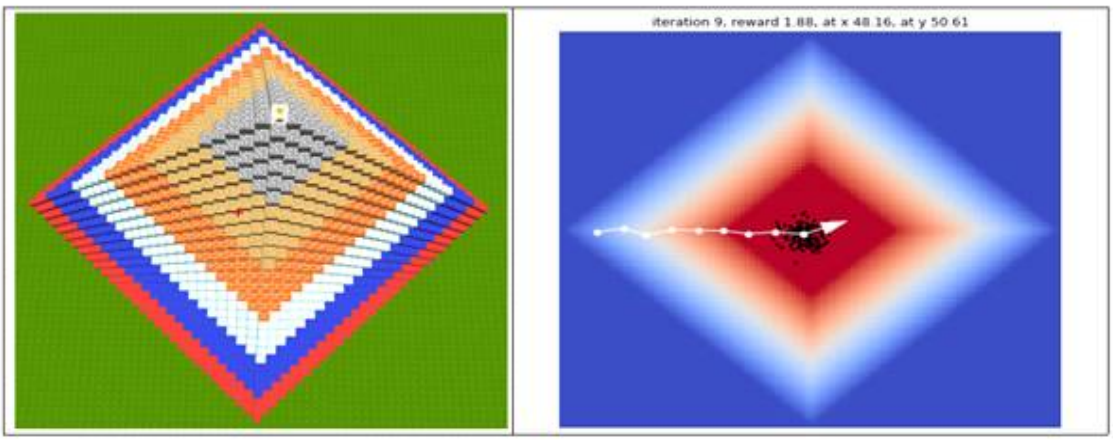
$$\mathbb{E}_{\varepsilon\sim N(0,1)}[\varepsilon f(x+\sigma\varepsilon)] \approx \mathbb{E}\left[\varepsilon f(x)+f'(x)\sigma\varepsilon^2+\frac{f''(x)}{2!}\sigma^2\varepsilon^3\right]$$

← (ㄹ), (ㅁ) (ㅂ)

$$= \mathbb{E}[f'(x)\sigma\varepsilon^2]$$

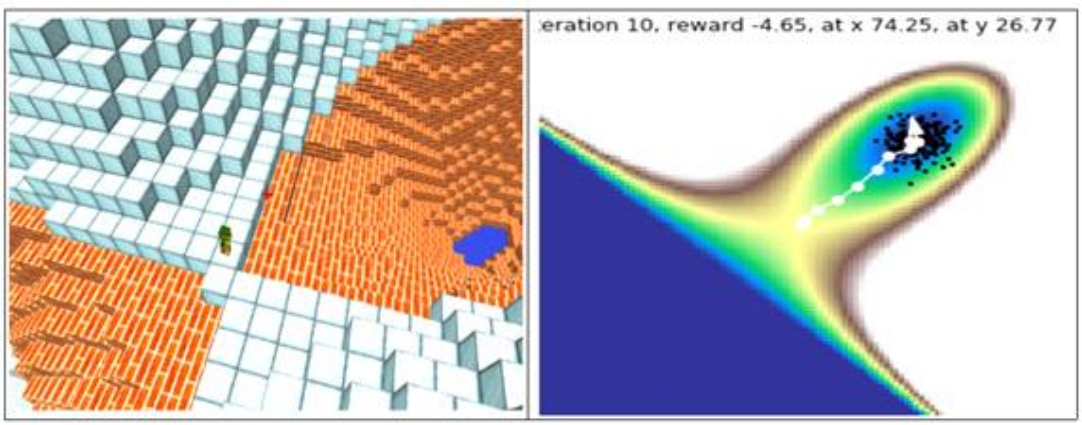
$$= f'(x)\sigma$$

$$\therefore f'(x) \approx \frac{1}{\sigma}\mathbb{E}_{\varepsilon\sim N(0,1)}\,\varepsilon f(x+\sigma\varepsilon)$$



터틀크래프트 인공지능 진화전략

$f(x,y)=x^3+y^3-3xy$ 의 함수 그래프가 그리는 지형에서 아래의 왼쪽에 보이는 터틀 거북이의 위치에서 오른쪽 호수로 내려가는 경로를 찾아보자. 여기에서도 미분이 되지 않는 경우에도 쓰일 수 있는 확률적 방법으로 아래로 가는 경사가 급한 곳을 따라 진화전략 알고리즘 (evolution strategy) AI 방법으로 찾는 것인데 앞에서도 말했듯이 진화알고리즘은 생물체가 환경에 적응하며 적자생존하는 진화의 법칙을 인공지능의 세계에 응용한 것이다.

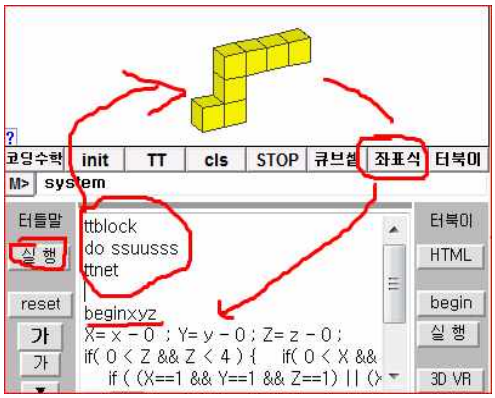


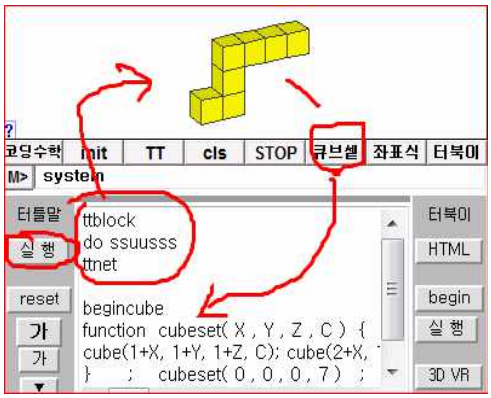
터틀말 쌓기나무 명령과 큐브식 좌표식의 창의융합 코딩

snucod.pythonanywhere.com/turbooki (임시로 간단하게 hanaone.com) 주소로 TurtleMAL40.zip 풀면 나오는 파이어폭스시작으로 가서 [터틀밀] 단추를 누르면 아래와 같은 명령을 할 수 있는 터틀말 및 터틀크래프트 코딩환경을 만날 수 있다.

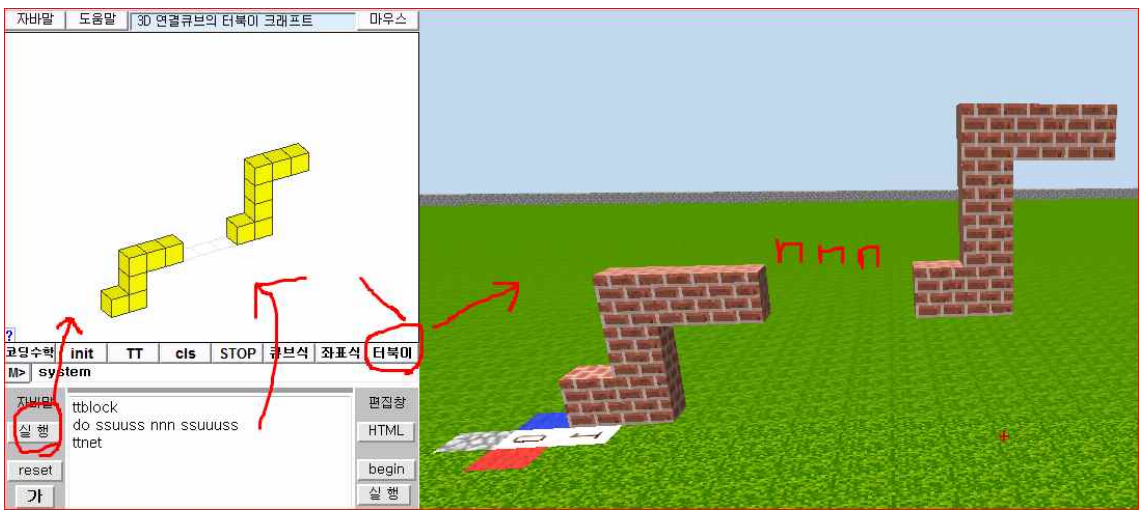
ttblock	쌓기나무준비
	ㄱ
do ssuusss	하자 ssuusss
ttnet	ㄴ

위 명령이 만든 쌓기나무의 좌표식 (왼쪽 xyz식) 과 큐브셸 함수 (오른쪽 cube셸)

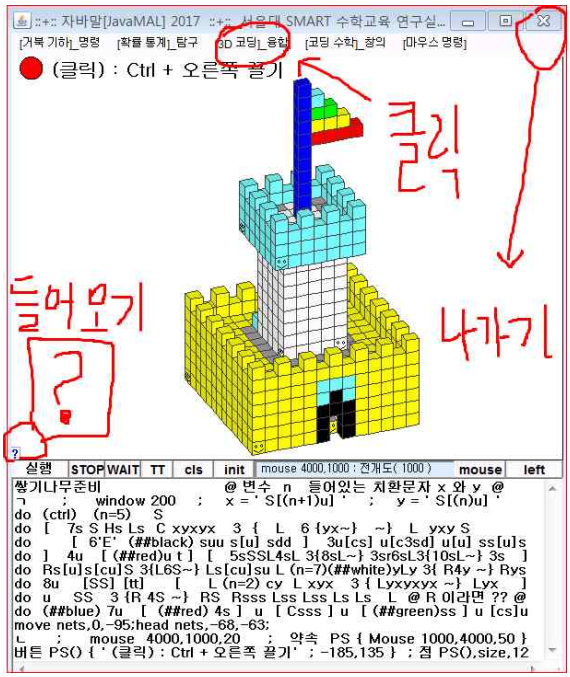




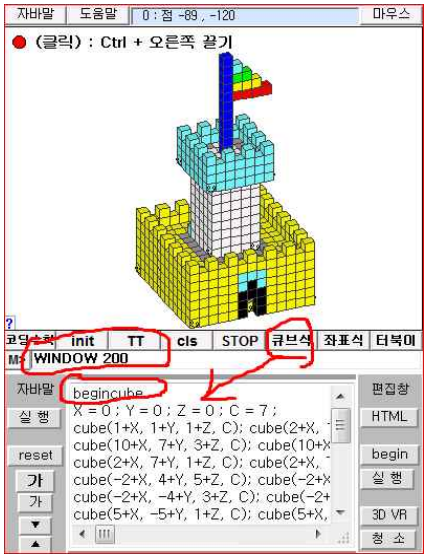
*** 예전에 자바매스 서버에서 가능했던 아래와 같이 터틀크래프트와 터틀말이 융합된 코딩환경을 snucode 에서도 가능하도록 만들 예정입니다.



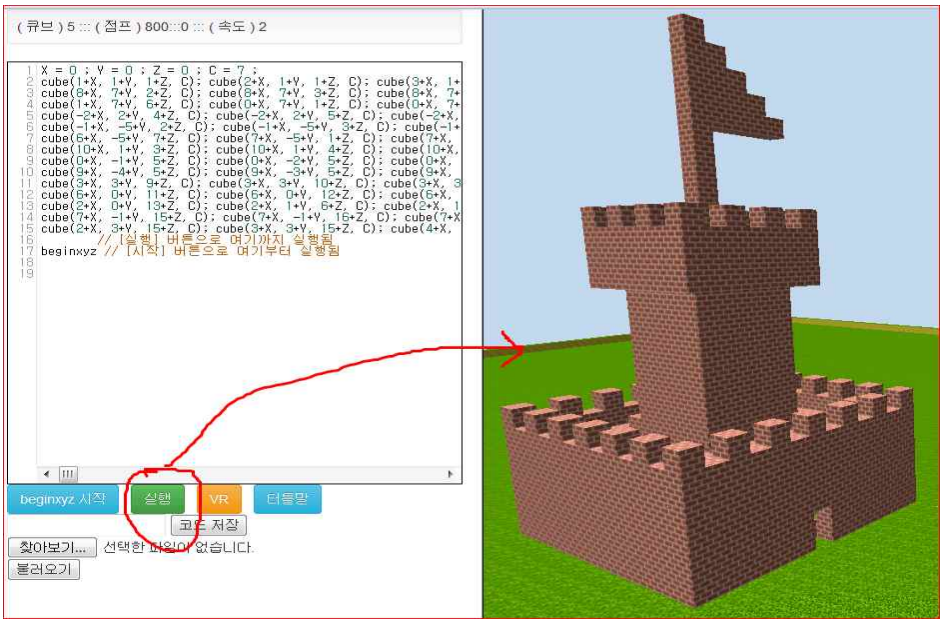
터틀말 화면의 [?] 단추를 누르면 메뉴가 있는 아래 왼쪽과 같은 편집기 창이 나타납니다. 이곳에서 [3D 코딩] 융합 메뉴를 눌러 레고성 명령을 불러서 레고 성을 그리고, [x] 단추로 나간 후, 아래 오른쪽 같이 [큐브식] 단추로 큐브식을 얻는 과정이 아래와 같습니다.



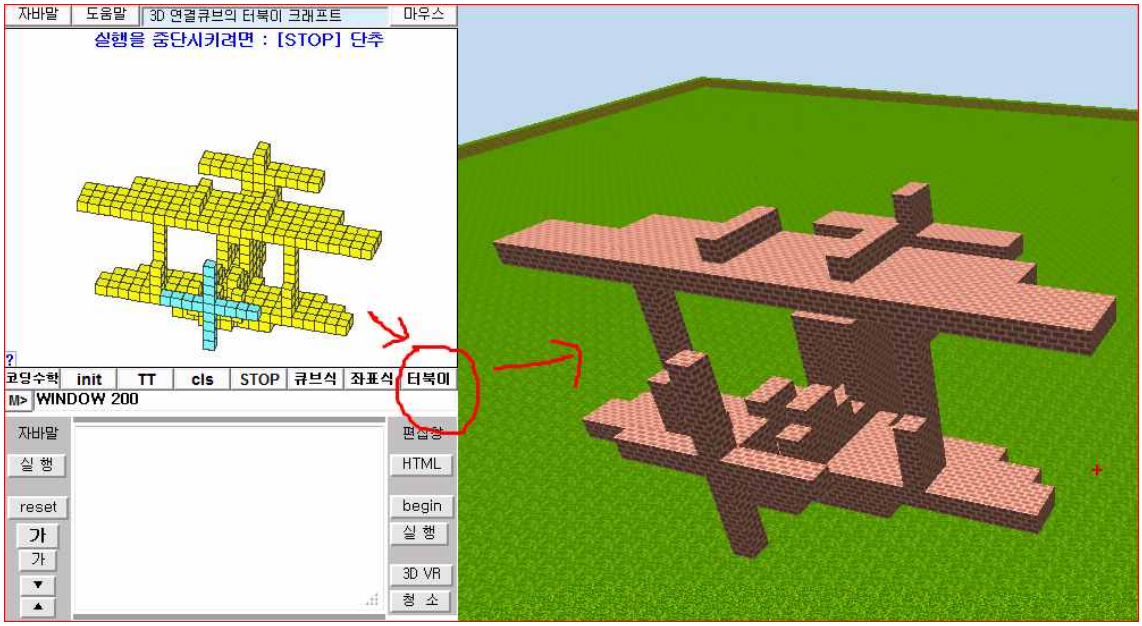
[?] 큰 편집기 창에 들어간다
[X] 작은 폅집기 창으로 되돌아 간다



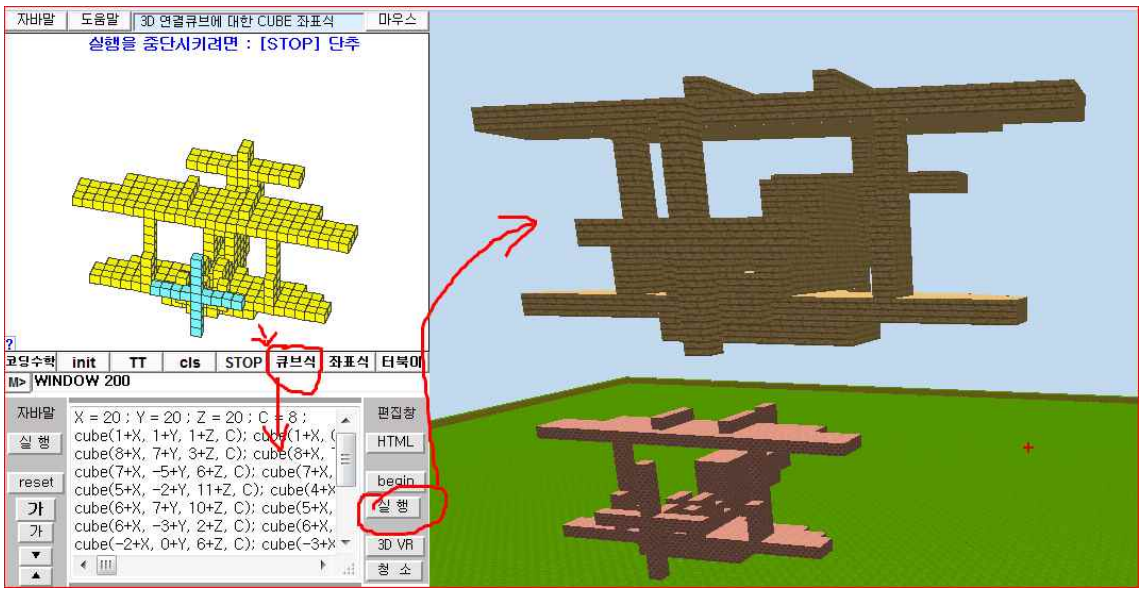
예를 들어, 레고성을 만든 후 [큐브식] 단추로 cube 좌표식을 만든 후, 터북이 크래프트 편집기에 복사하고 실행을 시키면 아래와 같이 나타난다 (begincube 는 지운다)



(1) 터틀말 화면에 [터북이] 단추가 있는데, 이 것을 누르면 오른쪽에 터북이 크래프트 화면이 생기면서 터북이가 터북이 크래프트에 마인크래프 같은 그림을 그린다



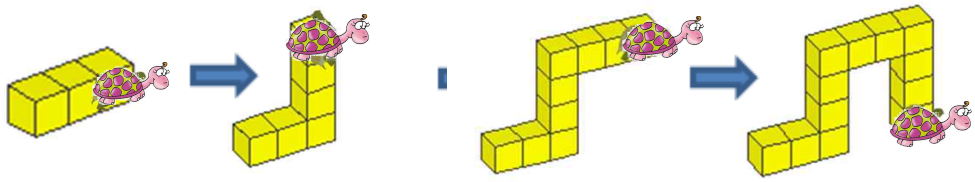
(2) 아래 그림은 위의 (1)에서 비행기를 그린 후, [큐브식] 단추로 비행기에 대한 큐브식을 만든 후, 이 것을 터북이 크래프트의 [실행] 단추로 실행을 시키면 크래프트에 그려진다.



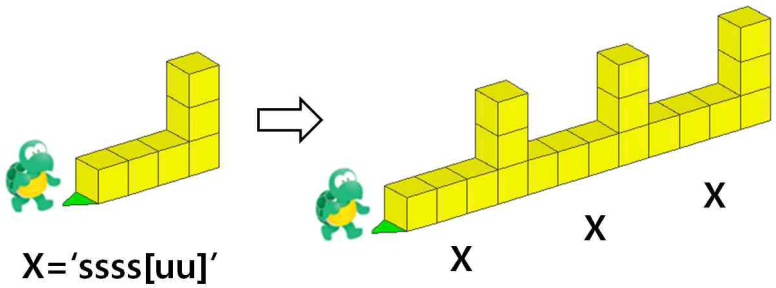
터틀말 거북명령으로 만든 비행기를 터북이 크래프트의 [beginzyz 시작] 단추와 [실행] 그리고 [3D VR] 단추 등으로 각각 실습하자. 터틀말과 터북이 크래프트가 연동되어 있음을 실습하자.

그런데, [터틀말] 사용법은 무엇인가요 ? 간단한 설명이 다음 페이지에 !!

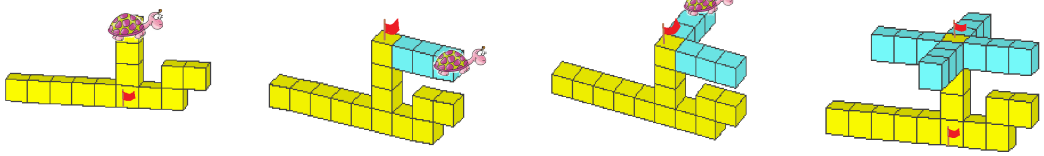
(1) (2) (3) (4)



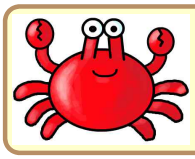
(1) do sss (do 3s)	(3) do sss uuu sss (do 3s 3u 3s)
(2) do sss uuu (do 3s 3u)	(4) do sss uuu sss ddd (do 3s 3u 3s)



(1) (2) (3) (4)

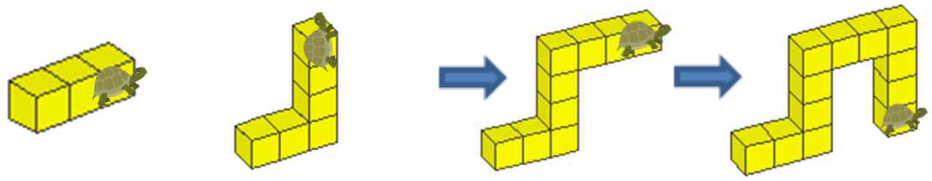


(1) do 6s [uuu] ss us	(3) do 6s [uuu [c3s] L [c3s]] ss us
(2) do 6s [uuu [c3s]] ss us	(4) do 6s [uuu [c3s] L [c3s] L [c3s] do L [c3s]] ss us



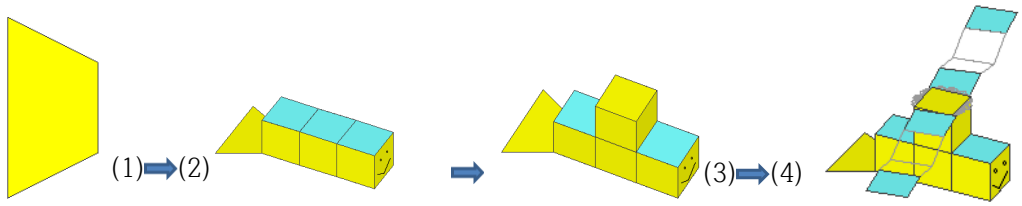
대문자 L (left) 명령으로 왼쪽으로 90도 돌고, r (Right의 r) 명령은 거북 진행 방향을 바꾸지 않고 게 처럼 옆걸음으로 오른쪽으로 이동. 대문자 R과 소문자 l 은 무엇일까? s 는 앞, t 는 뒤로

(1) (2) (3) (4)



(1) do sss (do 3s)	(3) do sss Usss Dsss (do 3s U3s D3s)
(2) do sss uuu (do 3s 3u)	(4) do sss Usss Dsss ddd (d와 D는 다름)

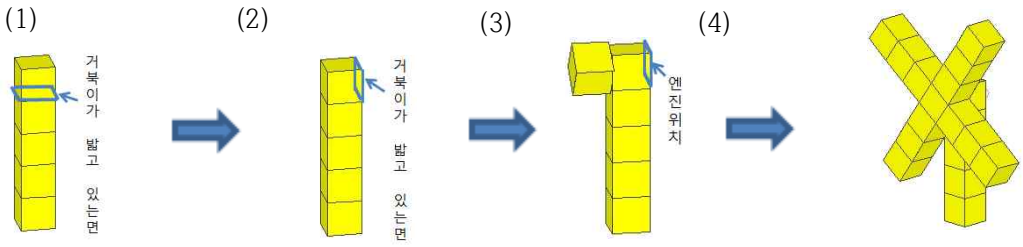
엔진 및 스프링 거북명령 : E 와 e



(1) do C(FF=5)(ffv=5)m 처음 길이는 10 인데, 폭 FF는 5이고 길이 ffv는 5인 면 m을 만듦	(3) do tu UU 10E t 하고, u로 위에 만들고, UU로 네 발을 천장에 놓아 천장에 달린 후, 10바퀴를 도는 엔진 E 만듦
(2) do (hhh=5) SSS FF=5, ffv=5 에 높이 hhh 까지도 5인 정육면체 SSS를 만듦	(4) do c [mem] LL men c 로 하늘색 만들고, [mem] 으로 스프링 e로 면 m과 m을 연결하고, LL 하고 mem 으로 또 만듦

약속 우 { do C(FF=5)(ffv=5)m(hhh=5); do SSStu UU10E c[mem]LLmem }

명령은 우주선을 만드는 명령을 약속 문자 '우' 에 기억시키라는 것이다. 여기서 **10E** 명령은 왼쪽 사진과 같이 현재 거북이가 발을 대고 있는 면 바로 위에 빙빙 도는 엔진 면을 만들고 거북이가 그 위로 올라가라는 것이다. **e** (또는 한글 ㅅ)는 2개의 면으로 접어지고 펼쳐지는 스프링을 만든다.



(1) do suuu	(3) do suuu U E u (엔진을 달고 u 만듦)
(2) do suuu U (U로 전환)	(4) do suuu U E u [3s] [L3s] [LL3s] [R3s]

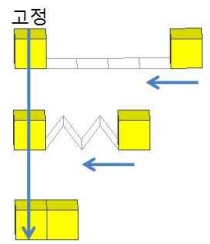


그림 (1)

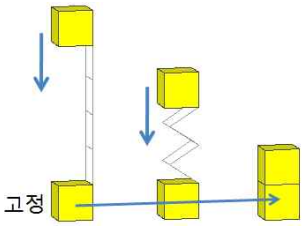
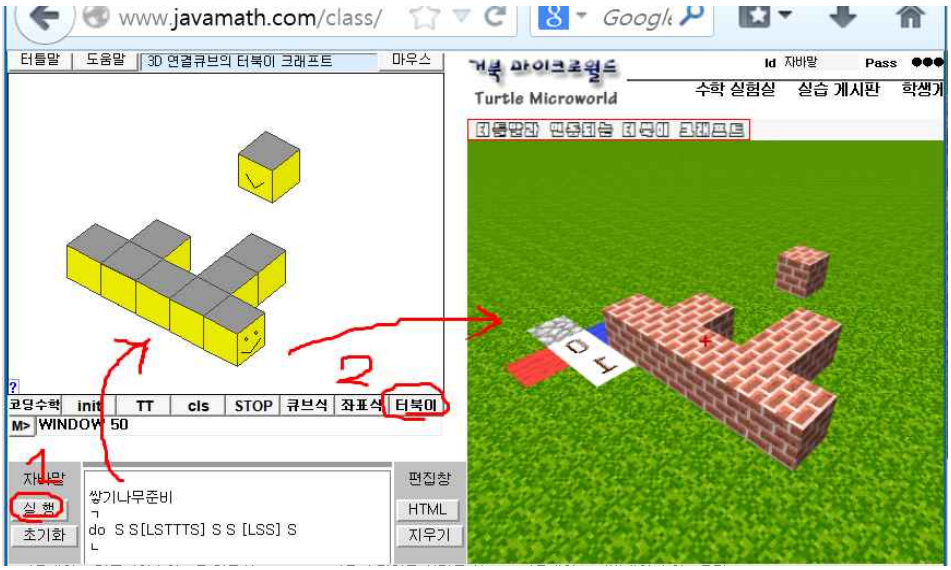


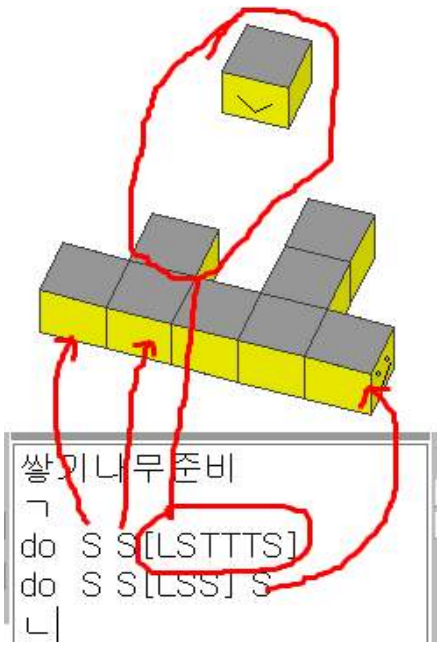
그림 (2)

do s eㅅ s (ㅅ 스프링 기호)	do s U ee s (또는 do sUㅅㅅs)

터틀말과 터북이 : 창의융합 미로 게임코딩 기본명령

위의 [터틀말] 단추를 누르면 나오는 [터틀말] 입력창에 아래 화면과 같은 명령을 쓰고, 아래와 같이 화면 왼쪽의 [실행] 단추를 누르고, 다음 [터북이] 단추를 눌러보자.





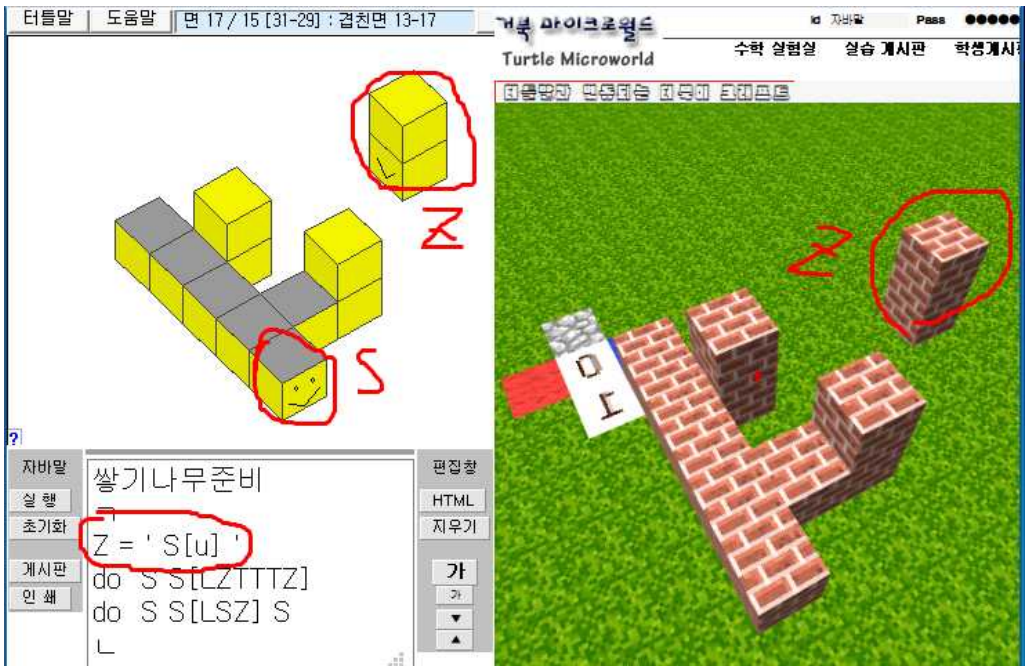
1. 쌓기나무준비 및 ㄱ 은 거북이 준비 (ttblock)

2. do S S[LSTTTS] 는 먼저 S를 만들고, 다음 S를 만들고, [에 의해 현재 거북이의 위치와 방향을 기억한 후, [다음에 있는 L 에 의해 왼쪽으로 90도 돌고, S와 투명한 쌓기나무를 3개 만드는 TTT 실행하고 S를 만든다. 마지막의] 기호는 [에 의해 기억되었던 위치와 방향으로 돌아간다.

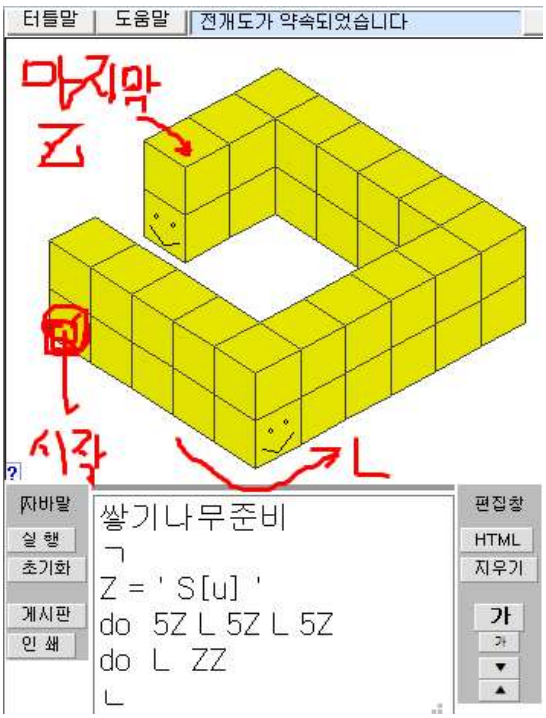
3. 이어서, 그 다음 줄의 do S S[LSS] S 가 실행되어, 먼저 S 만들고 그 다음 S[LSS] 하고 S를 마지막에 만들어 화면과 같은 그림을 만든다.

4. 대문자 S 는 소문자 s와 같이 쌓기나무를 만들지만, 얼굴과 꼬리 등 모양이 같이 그려지게 만든다. 꼬랑지 그림과 얼굴 그림으로 방향을 파악한다.

미로를 만들려면 2층으로 벽을 만들어야 의미있는 미로게임을 만들 수 있다. 2층으로 쌓기 나무를 쌓으려면 S[u]를 하면 된다. 즉 먼저 S를 만들고 [으로 그 자리를 기억 후, 위로 올라가 u를 만들고, 다시] 에 의해 아래로 내려오면 된다. 이제 S[u]를 대문자 Z라고 약속을 하고 명령을 쓰는 치환명령 기법을 알아보자.



위 화면의 편집기에 쓰인 Z = ' S[u] ' 는 앞으로 Z를 실행하라고 하면 S[u]를 하라는 것이다. 따라서, do S S[LZTTTZ] 하면 Z가 쓰인 곳에 S[u]가 실행되어 이층이 만들어진다.



1. 왼쪽은 Z = ' S[u] ' 로 약속을 하고, do 5Z L 5Z L 5Z 명령 등으로 2층 벽을 만드는 모습이다.

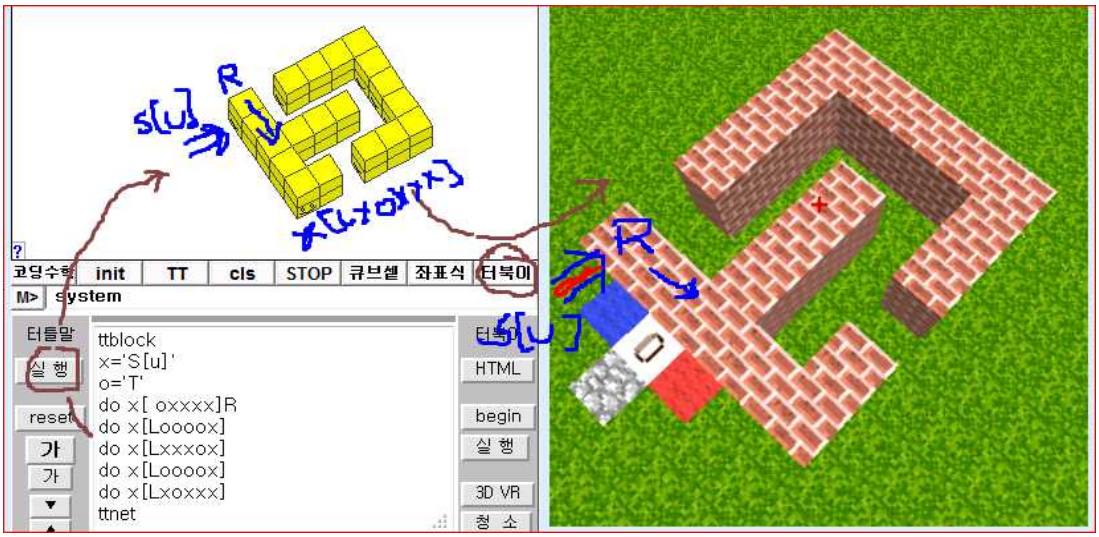
2. 만일 Z = ' S[uuu] ' 로 약속을 하고 실행시키면 어떻게 될까 ? Z = ' S Z ' 라고 약속을 하면 무슨 일이 벌어질까 ??

3. Z = ' S[u] ' 명령으로 Z에 명령 S[u]를 기억시키는 것을 수학교과서의 용어를 따라 치환한다고 하고, Z를 치환 문자라 한다.

4. 치환된 내용을 지우려면 새로 Z에 치환을 하거나, [실행] 단추 밑에 있는 [초기화] 단추를 누른다. [초기화] 단추는 터틀말을 처음 시작한 초기 상태를 만들어준다.

터틀말과 터북이 : 창의융합 미로게임 코딩컨테스트

아래의 ttblock ~ ttnet 미로 만들기 명령에서, x = ' S[u] ' 는 문자 x에 명령 S[u]를 기억시키는 즉 대입시키는 명령입니다. 대문자 S는 소문자 s 와 같이 쌓기나무를 만들지만 거북이의 얼굴과 꼬리가 있습니다. 대문자 T는 투명한 쌓기나무를 만들고 o = ' T ' 입니다. [와] 는 현재 위치와 방향을 기억하고 다시 돌아가게 하는 명령이고, L은 왼쪽으로 90도 그리고 R 은 오른쪽으로 90도 터북이가 몸의 방향을 전환하는 명령입니다.



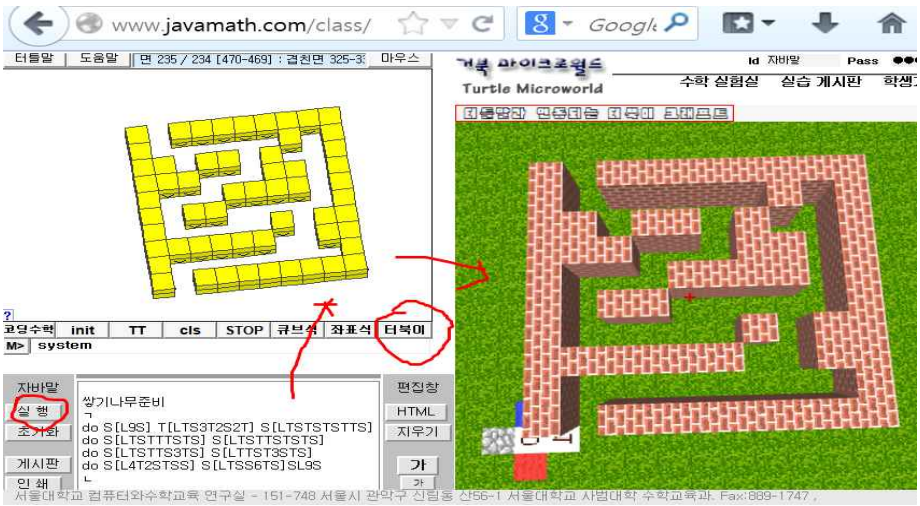
(설명) 위는 터틀말에서 ttblock ~ ttnet 사이에 o와 x로 미로 게임을 만들고, [터북이] 단추로 실행시킨 모습이다. 아래는 [큐브셀] 단추로 만들어진 큐브셀을 터북이 크래프트 편집기에 쓰고, [beginxyz] 단추로 풀밭을 만든 후 (아무 명령도 없는 beginxyz 명령은 디폴트로 아무것도 없는 풀밭을 만듦), [실행] 단추로 미로 게임을 나타나게 만든 화면의 모습.

[큐브셀] 으로 만들어지는 아래 왼쪽 명령과 [큐브셀]으로 만들어지는 아래 오른쪽 명령은 똑 같은 기능을 한다. 오른쪽과 같이 큐브식이 만드는 명령을 { 과 } 으로 감싸며 모은 cubeset 함수로 만들고 cubeset(0, 1, 2, 7) 로 X=0 ; Y=1 ; Z=2 ; C=7 역할을 한다.

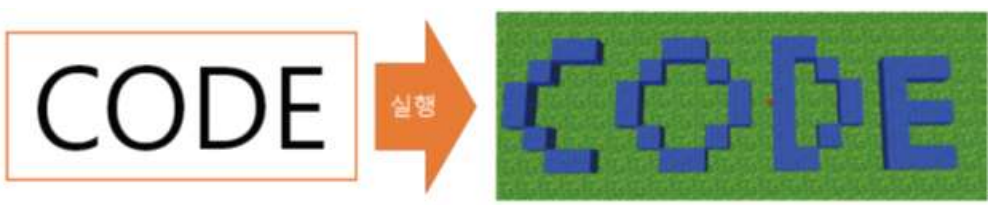
```
@ var maze = [
    "# ########",    // WASD 키명령
    "#       # #",    // Z : 위로
    "## ## #  #",    // X : 점프+앞
    "###   ## #",    // C : 아래로
    "#   #### #",    // Shift : 중력
    "# ##     #",    // Space : 점프
    "#     # ##",    // BNM :점프 높이
    "###### ##",     // KL : 속도 조절
    "#       #",    // GH : Go Home
    "# ########"     // 화살표키 사용
];     // Ctrl+마우스, Alt+마우스 블럭 @

쌓기나무준비
ㄱ
S = ' S[u] '
do S[L9S] T[LTS3T2S2T] S[LTSTSTSTTS]
do S[LTSTSTTSTS] S[LTSTTSTSTS]
do S[LTSTTS3TS] S[LTTST3STS]
do S[L4T2STSS] S[LTSS5TS]SL9S
ㄴ
```

[다른 방법으로 미로 게임 만들기]
자바스크립트 기반의 터틀 크래프트에서 만든 미로게임 형태가 var maze = [미로 모양] 안에 쓰여 있다. 이 것을 터틀말 쌓기나무 코딩으로 바꾸려면, 미로의 왼쪽 아래를 시작점으로 하여, 쌓기나무준비 ㄱ ㄴ 사이의 명령을 한다
[명령 설명] do S[L9S] 와 같이 먼저 S로 나가고, [으로 현재 위치와 방향을 기억하고, L에 의해 왼쪽으로 90도 돌아서, 9S로 9개의 S를 만들고,] 에 의해 다시 원위치로 돌아오라는 형태의 명령을 반복하여 사용하여 미로게임의 모양을 그린다.
[치환 명령] S = ' S[u] ' 명령에 의해
do S 를 시키면 S가 기억하고 있는 S[u] 가 실행되어, S를 만들고 그 위에 u를 하나 더 만들고 다시 아래의 위치로 돌아온다. 이렇게 하면 높이가 2층인 미로가 만들어진다.

[참고] 위의 명령에서 @ var maze ..... @ 뜻은 @ 와 @ 사이의 명령은 터틀말에서 실행하지 않는다. 참고로 쌓기나무준비 ㄱ 명령은 영어로 ttblock 이고 ㄴ 명령은 ttnet 이다.



[참고] 미로 게임을 만들고, 아래와 같이 자기 이름이나 글자를 만들 수도 있다. 그리고 VR 단추를 눌러 3D VR 로 만들고, 놀고, 같이 나누는 Making, Playing, Sharing 코딩을 한다.



말게임(번호) 및 말게임 만들기

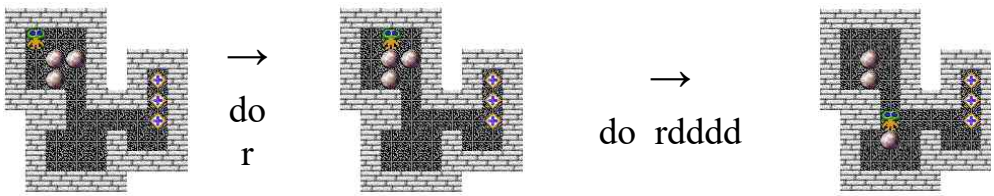
터틀말에는 미로 게임보다 한단계 위인 푸쉬푸쉬 (소코반) 게임을 만들고, 또한 답을 제시할 수 있다. 또한 이미 만들어져 있는 105개의 게임을 할 수 있다.

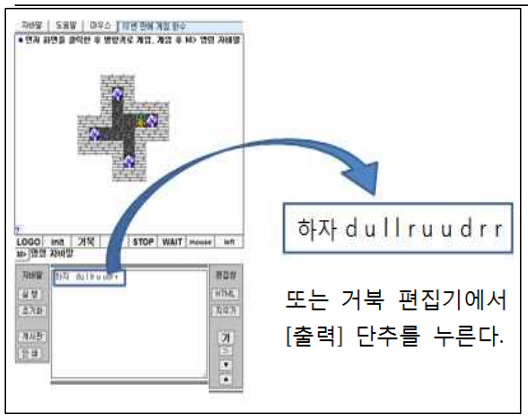
바로 왼쪽은 어려운 104번 푸쉬푸쉬 게임으로 M> 말게임(104) [엔터] 하여 게임을 불러낼 수 있다.

아래는 편집기 명령으로 게임 만드는 모습

```
약속 말게임 { ;;;;
   nnnnnnn ;  n++o++n ;  n++n++n ;  n_ooo_n ;
   n__o__n ;  n_ooo_n ;  n__n%_n ;  nnnnnnn
}

말게임 ; pause 100

do ruulur dddl uruuuu llldlddru
do uddl ddruuu ddrrdr uuuu dl
do urdlll dludd rrruu lludrrdruu

@ 약속 말게임 { 와 } 안에 위와 같이 @
@ 게임을 만들고, 왼쪽 같이 말게임 @
@ 하여 게임을 부르고, pause 후 게임 @
@ n 벽; o 공; + 공 놓을 곳, % 무주인 @
@ _ 바닥면; & 는 + 와 o 동시인 상태 @
@ ; 다음 칸으로, 스페이스는 빈공간 @
@ do 명령 ==> l, r, u, d 키보드 @
@ M> 말게임(20) [엔터], 화살표 키로 게임하고 [명령] 눌러 게임을 저장 @
@ 위 설명을 따라 말게임 (거북 창의게임)을 만들고, 게시판에 저장시킨다 @
```

위(u), 아래(d), 왼쪽 (l), 오른쪽 (r) : 푸쉬푸쉬 거북게임



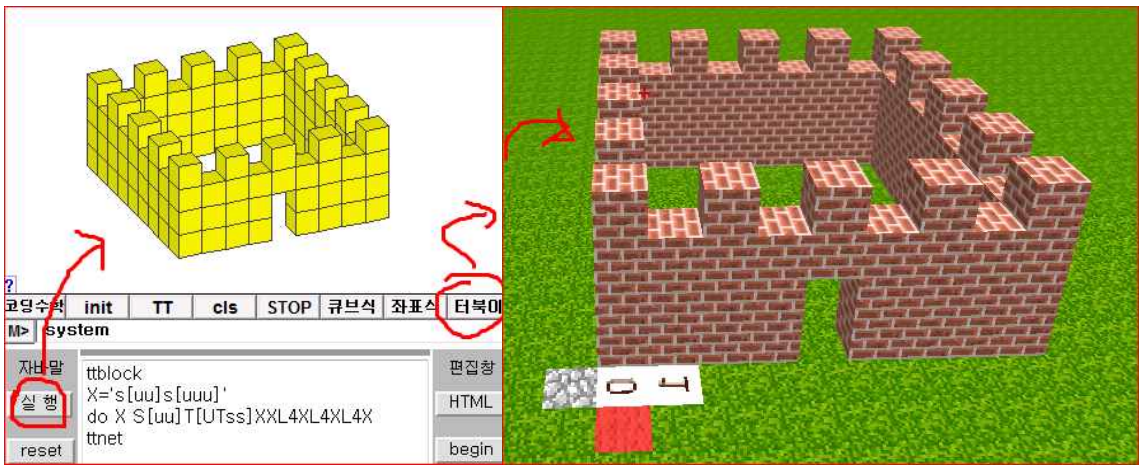


또는 거북 편집기에서 [출력] 단추를 누른다.

거북명령 푸쉬푸쉬 게임을 하려면 화면 아래의 입력창 M> 에 M> 자바말(1) [엔터] 과 같이 자바말(게임 번호)를 쓰고 엔터한 후, 화살표 방향키 4개로 게임을 하는데, 자바말(0) 부터 자바말(104) 까지 105 가지 게임이 가능하다. 게임을 마친 후, M> 명령 자바말 [엔터] 하면 왼쪽과 같이 편집기에 게임 번호 자바말(1) 과 do dudr .. 과 같은 거북명령이 쓰여진다. 이를 복사해 게시판에 저장하고, 나중에 실행시키면 게임하는 장면을 동영상으로 누구나 볼 수 있다.

터북이 크래프트와 터틀말을 활용한 나만의 object 만들고 심기

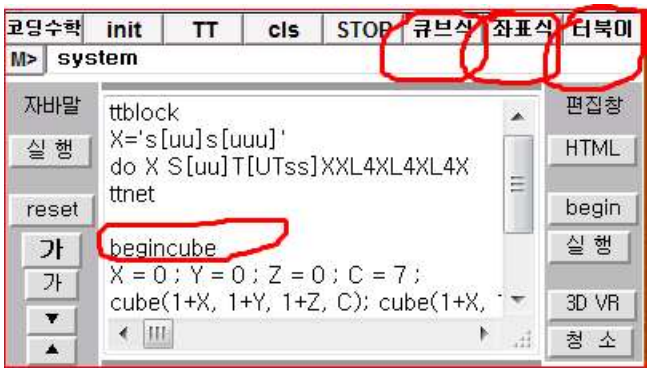
위의 [터틀말] 단추를 누르면 아래와 같은 화면이 나옵니다. 편집기에 아래와 같이 명령을 쓰고 터틀말의 [실행] 단추를 눌러 실행시키면 아래와 같은 레고성의 모양이 그려집니다. 안되는 경우, 파이어폭스의 부가기능 옵션에서 [Java 6.25] 를 [항상 사용] 으로 합니다.



[명령 설명] ttblock 과 ttnet 사이에 쌓기나무 명령을 쓰고, [실행] 단추로 실행시킵니다.

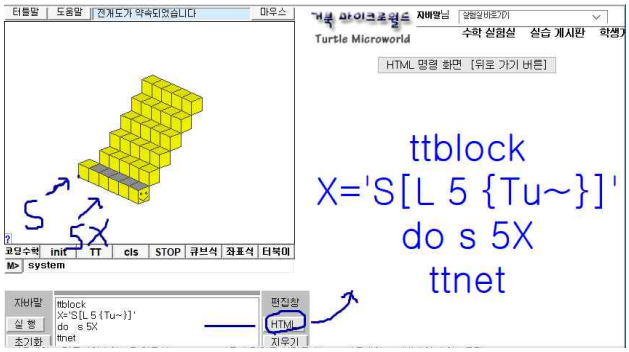
**** X = ' s[uu]s[uuu] ' 와 같이 따옴표 사이에 명령을 쓰고 문자 X에 명령을 저장할 수 있습니다. 4X 는 X가 네 개인 XXXX 를 나타내며,
**** 대문자 U는 정육면체 안에서 터북이가 앞의 면에 발을 대고 위를 보며 서게 됩니다. 그 후, T 하면 투명한 (Transparent) 정육면체를 만듭니다. 대문자 S와 소문자 s 모두 정육면체를 만들고, [와] 은 도돌이표처럼 각각 위치를 기억하고 다시 그 위치로 돌아가게 합니다. 소문자 u와 d는 각각 위와 아래에 정육면체를 만듭니다.

**** 이제 편집기 창에 명령을 쓰고 [실행] 단추를 누르면 위와 같이 성 모양이 나옵니다.
**** 그 후, [터북이] 단추를 누르면 오른쪽과 같이 터틀크래프트에 성 모양이 그려지게 됩니다. 이제 [마우스]로 터틀 크래프트 화면을 클릭하고, [포인터 숨기기]를 하여 터북이의 세상으로 들어가서 갑니다.
**** 터북이 세상에서 마우스로 터북이의 시선을 바꿀 수 있으며, W, A, S, D 는 각각 앞으로, 왼쪽으로, 뒤로, 오른쪽으로 터북이를 움직입니다. Z키는 터북이를 위로, C키는 터북이를 아래로, [Shift] 키는 중력을 작동시켜 터북이를 땅으로 떨어지게 합니다. 스페이스 키는 거북이를 깡총거리게 만들며, X는 터북이를 깡총거리며 앞으로 나아가게 합니다.
**** [Alt] 키를 누른 상태에서 마우스로 블록을 클릭하면 블록이 깨어집니다. [Ctrl]키를 누른 상태에서 마우스를 클릭하면 구 위치에 블록이 생깁니다.



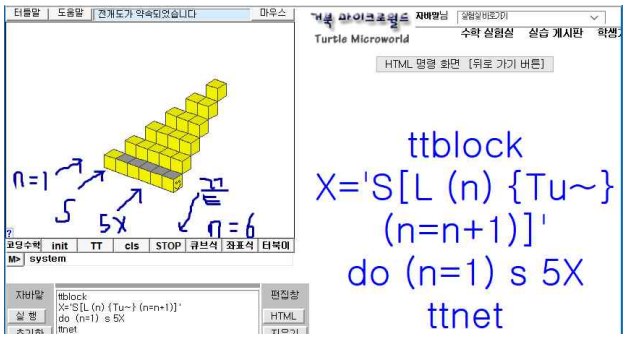
거북 명령으로 그림을 그리고, [큐브식] 을 누르면 왼쪽과 같이 begincube 라고 쓰인 곳 밑에 큐브식이 쓰입니다. 큐브식만 복사해서, beginxyz 위에 씁니다. 이 때, 대문자 X, Y, Z, C 와 for 반복 명령을 잘 활용하여 그림을 만들어본다 (큐브셀 대신에 큐브식 씀).
[정리] cube(a,b,c,d) 는 좌표 (a,b,c) 에 d번 블록을 만듭니다.

치환문자의 특별한 명령 :



치환문자 X = ' S [L 5 { Tu ~} ' 일 때, do s 5X를 실행시키면
먼저 s 가 실행되고, 다음 XXXXX 가 실행된다. 각각의 X는 먼저 S가 그려지고, [으로 현재 위치와 방향이 기억되고, 5 { Tu ~} 명령은 { 와 ~} 사이에 묶여있는 Tu를 다섯 번 반복해서 실행시킨다. 마지막으로] 에 의해 원위치한다.

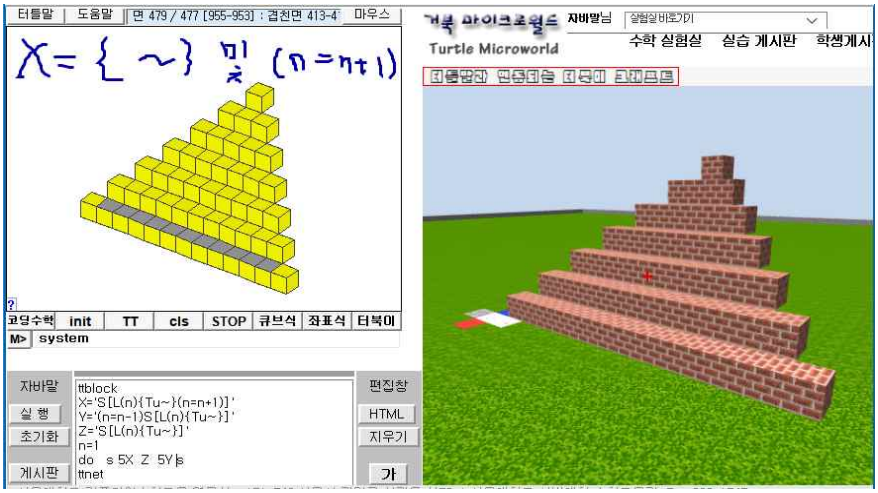
(주의) do 5 T u 와 do 5 { T u ~} 명령의 차이가 무엇인가 ?
{ 과 ~} 사이의 명령은 하나로 묶이기에, 5 { Tu ~} 하면 Tu Tu Tu Tu Tu 가 실행된다.



X = ' S [L (n) { Tu ~} (n=n+1) ' 일 때, do (n=1) s 5X를 실행시키면
먼저 (n=1) 이 실행되어 변수 n의 값이 1로 고정된다. 다음 s가 실행되고, 5X 즉 XXXXX 가 실행된다.
현재 n의 값이 1 이기에 처음 X에 의해 S[L 1 { Tu ~} 가 먼저 실행되고 다음의 (n=n+1)] 이 실행된다.

(n=n+1) 의 명령은 현재의 n 값을 하나만큼 증가시킨다. 따라서, 현재의 n의 값이 1 이기에 n이 2가 된다. 두 번째 X 가 실행되면 S[L 2 { Tu ~} (n=n+1)] 이 실행되고, 마지막에 (n=n+1) 에 의해 n의 값이 3이 된다. 세 번째 X가 실행되면 S[L 3 { Tu ~} (n=n+1)] 가 실행되고, 마지막에 n의 값은 4가 된다. 따라서 4번째 X 가 실행되면 S[L 4 { Tu ~} (n=n+1)] 이 실행되어 n이 5가 되고, 마지막 5번째 X가 실행되면 S[L 5 { Tu ~} (n=n+1)] 가 실행되어 에 S[L Tu Tu Tu Tu Tu (n=n+1)] 명령이 실행되는 결과가 되고, 마지막의 (n=n+1) 에 의해 최종적으로 n의 값이 6이 된다.
((탐구 문제)) X = ' S [L (n) { Tu ~} (n=n-1) ' 일 때, do (n=5) s 5X 하면 ????

피라미드 탐구

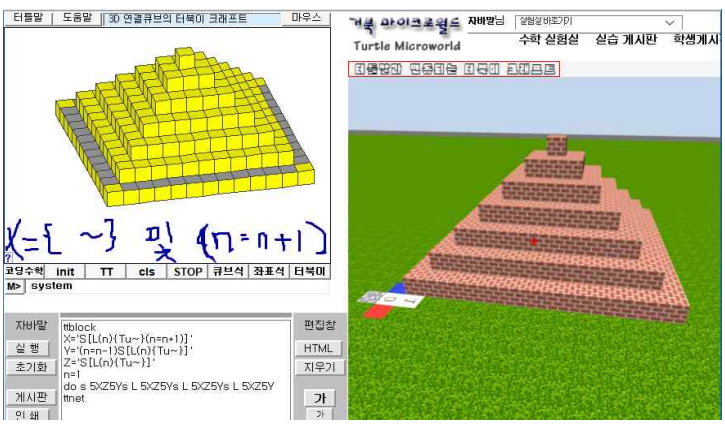



```
ttblock

X='S[L(n){Tu~}(n=n+1)]'
Y='(n=n-1)S[L(n){Tu~}]'
Z='S[L(n){Tu~}]'
n=1
do    s 5X  Z  5Y s

ttnet
```

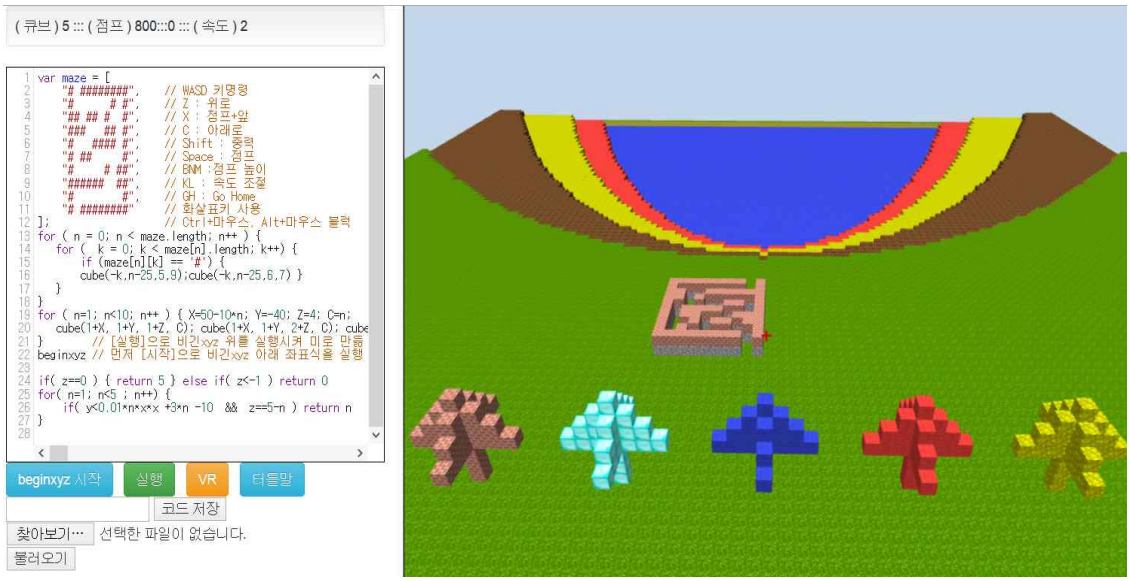
(명령 설명) 치환문자 안에 { ~} 과 (n=n+1) 사용
X = ‘ S [L (n) { Tu ~} (n=n+1)] ’ 로 지환하고 변수 n=1 인 상태에서 do X X 하면 어떻게 될까 ?
(설명) 먼저 처음 X 즉 S [L (n) { Tu ~} (n=n+1)] 가 실행되어, 먼저 S가, [으로 위치 방향 기억, 그 다음의 L 로 90도 왼쪽으로 회전, (n) 의 현재 n의 값 즉 1 이다. { 와 ~} 기호의 그 안의 명령 Tu를 묶는 명령으로, 예를 들어, 4 { Tu ~} 는 TuTuTuTu 와 같다.
마지막의 (n=n+1) 은 n의 값을 하나 증가시킨다. 따라서 2가 된다. 그 다음의 X가 실행되면, n의 값이 2이기에 (n) { Tu ~} 는 TuTu 와 같고, 마지막의 (n=n+1) 에 의해 n의 값이 3이 된다.

[탐구 문제] 위 명령과 같이 치환문자 X, Y, Z 하고 n=1 인 상태에서 do s 5X Z 5Y s 하면 피라미드의 4면 중 한 면이 만들어진다. 이제 다음 명령을 한 결과는 무엇일까 ? 왜 ?

do s 5X Z 5Y s L 5X Z 5Y s L 5X Z 5Y s L 5X Z 5Y



수학과 코딩을 융합한 코딩수학으로 만들며 학습한다



[[코딩수학 탐구]] 위와 같이 편집기에 명령을 쓰고 먼저 beginxyz 아래에 쓰인 명령을 [beginxyz] 단추를 눌러 실행을 시킵니다. 그 실행된 모습이 피요르드 해안선의 모습입니다. 그 다음에 [실행] 단추를 누르면 beginxyz 위에 있는 명령들이 실행되면서 미로 게임을 할 수 있는 게임판과 나무 모양이 여러 군데 심어지게 됩니다. 여기 코드를 활용하여 수학과 코딩이 만나는 코딩수학 탐구를 시작합니다.

화면에 들어가서, 마우스를 좌우로 회전시키면 (마우스를 클릭하지 날고 그냥 움직입니다) 3차원 화면이 나타납니다. 마인크래프트 게임에서와 같이 W (앞으로), A (왼쪽) , D (오른쪽) , S (뒤) 로 터북이가 움직입니다. 터북이는 크래프트 화면에 사는 거북이 이름입니다.

W, A, S, D	이동	마우스 왼쪽 버튼 클릭	
마우스 이동 또는 방향키	시점 변환	Ctrl + 마우스 왼쪽 버튼 클릭	
Space Bar		Alt + 마우스 왼쪽 버튼 클릭	
Z, C		G	
Shift		숫자 키	

이제, Z 키를 누르면 터북이가 우로 올라가고, C 키는 아래로 내려갑니다. 이 때, 중력이 없어져서 턱북이가 날아다닐 수 있습니다 (위로 올라가서 W, A, S, D 키를 눌러보세요)

왼쪽 표를 기반으로, 키보드를 눌러보면서, 빈 공간에 해딩히는 기능을 쓰며 익혀보세요.

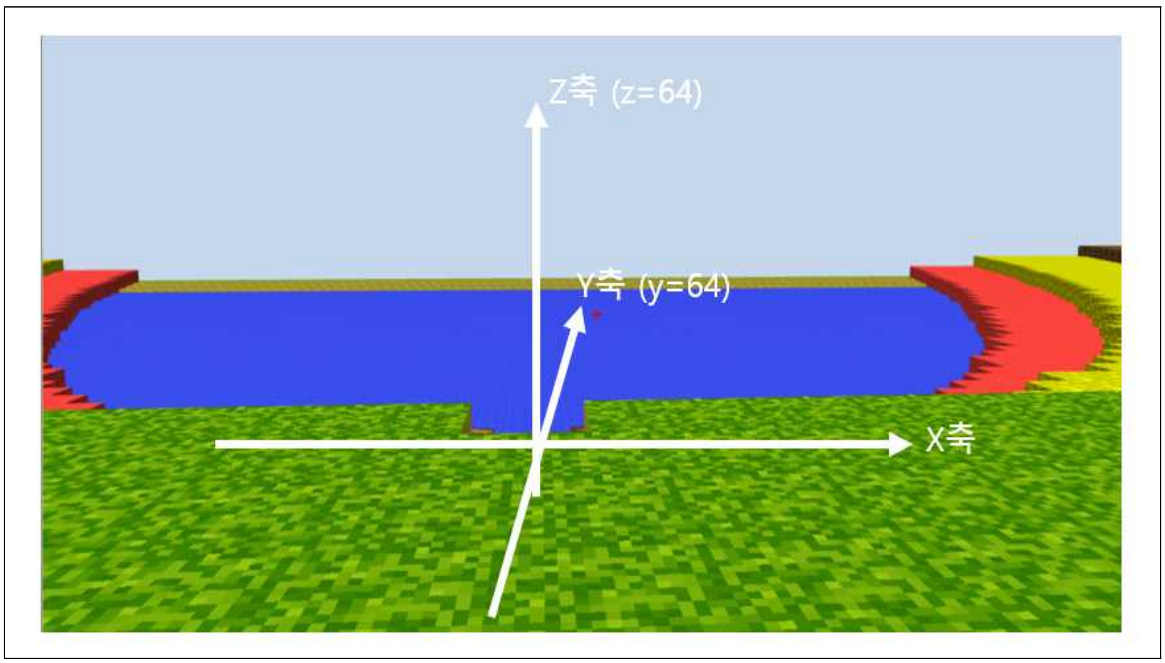
다시 중력이 작동하게 하려면 [Shift] 키를 누릅니다. 거북이가 땅으로 떨어지게 됩니다. 이제 Z 키로 하늘에 올라간 후, S 키로 뒤로 움직이며 터북이가 사는 동네를 바라보세요. 해안선을 따라 만들어진 방풍림을 살펴보세요.

(먼저) 편집기 명령 중에 beginxyz 아래에 쓰인 다음 명령을 봅니다. [beginxyz 시작] 단추를 누르면 beginxyz 아래에 쓰인 바로 아래의 명령이 실행되어 그림이 나타납니다.

```
beginxyz // 다음 [실행]으로 beginxyz 위의 실행식을 실행
         // 화면 클릭 및 ESC 키로 코딩화면을 전환 시킴

if(z<-1 || z>20) {return 0} else if(z==1) return 5
for( n=1; n<=5 ; n++) {
    if( y<0.01*n*x*x +3*n -10  &&  z==6-n ) return n
}
```

==========

1) 화면에 나타난 위치를 x 축, y 축, z 축을 통해 좌표로 알아봅시다.



터북이 크래프트 세상에서는 x, y 좌표는 원점을 기준으로 -63부터 +64 사이의 좌표값을 가질 수 있고, z축은 -64부터 +63까지의 값을 갖는다. [beginxyz 시작] 실행의 마지막에는 자동으로 z 축의 -64에 지옥으로 떨어지는 것을 방지하기 위해 return 9 으로 돌마당이 만들어지고, 0 층에 풀밭이 -1층에 돌마당이 만들어진다. 따라서 0 층의 풀밭 때문에 -64층의 돌마당이 안보일 수 있다. 이 때, **Alt+마우스 클릭으로 풀밭을 깨면 돌이 나오고 그 것을 깨면 -64 층의 돌마당이 보일 것이다. Ctrl+마우스 클릭은 블록을 만든다.**

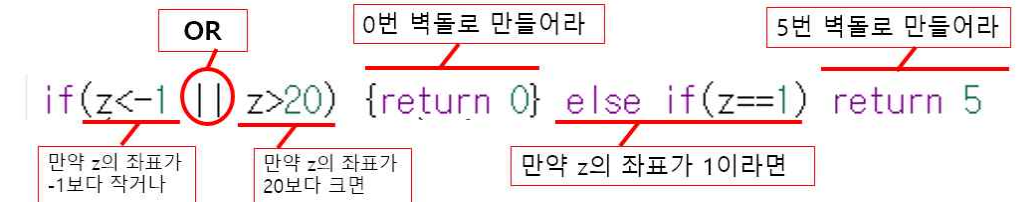
2) 기본 코드 분석

터북이 크래프트를 실행 시키자마자 나오는 기본 코드에 대해 공부하면서 어떻게 수학을 사용해야 터북이 코딩을 멋있게 할 수 있는지 생각해보겠습니다 (터북이는 화면 속 거북이).

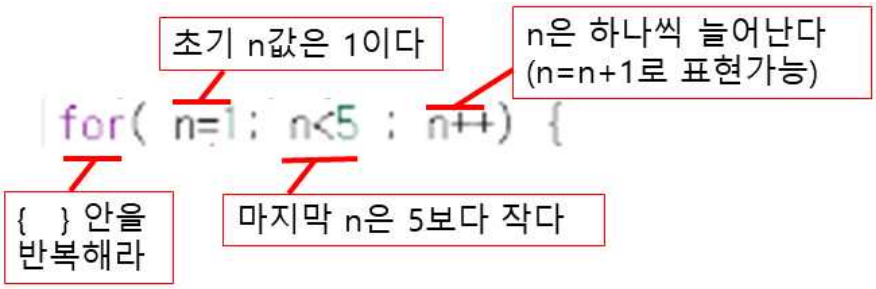
***** beginxyz 밑에 다음과 같은 코드가 있습니다. 1번 줄부터 확인해보겠습니다 ******
(참고) 아래에서 beginxyz 다음에 z<-1 이면 return 0 이라고 했기에, beginxyz 마지막에 자동으로 첨가되는 if(z==-64) return 9 명령이 실행되지 못해서 돌마당이 생기지 않는다.

```
1  beginxyz
2
3  if(z<-1 || z>20) {return 0} else if(z==1) return 5
4  for( n=1; n<=5 ; n++) {
5      if( y<0.01*n*x*x +3*n -10  &&  z==6-n ) return n
6  }
```

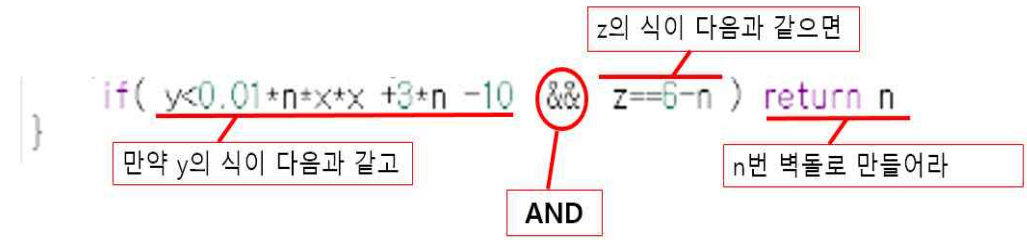
3



4



5



n이 하나씩 증가함에 따라 식이 어떻게 바뀌는지 적어봅시다! 그리고 말로 적어봅시다

참고! AND와 OR의 차이는 무엇일까요?? 터틀 코딩에서 명령어는 다음과 같아요.

AND	OR
&&	||

키보드에서 &는 숫자 7과 함께 있고 | 는 Backspace 밑에 원화모양(\)과 함께 있어요. & 명령어는 shift + 7 , | 모양은 shift + \을 함께 누르면 됩니다.

[탐구문제] 맨 처음에 있는 바로 아래의 명령을 그 아래와 같이 고치고 실행시켜 보세요.

```
if(z<-1 || z>20) {return 0} else if(z==1) return 5
```

```
if(z==63) {return 4} else if(z==1) return 5
```

두 경우의 실행되는 속도를 스톱워치로 나타나는 시간을 측정해봅시다. 왜 속도의 차이가 날까요 ? 맨 처음에 0 으로 채워지는 즉 블럭이 필요없는 곳을 빨리 거북에게 알려주면 거북이의 일을 대폭 줄여줄 수 있습니다. 거북이는 모든 곳의 좌표에 대해 점검을 합니다.

[탐구문제] 다음 두 명령의 차이점과 같은 점을 탐구하세요
beginxyx
if(y==x) return 7

이 명령은, 다음과 같은 알고리즘에 의해 128^3 만큼 일을하며 블럭을 만들며 작동합니다

```
function bb(x, y, z) {
  if(y==x) {  return 7 }  else {  return 0 }
}

for(var i=-63; i<=64; i++) {          // x 와 y 좌표의 최소 좌표는 -63
   for(var j=-63; j<=64; j++) {       // x 와 y 좌표의 최대 좌표는 64
      for(var k=-64; k<=63; k++) {  // z 축은 -64층에 지옥행을 막는 돌마당이 있다.
         b = bb(i,j,k)                  // 대신에 z 축의 최고 좌표는 하나 작은 63
         if(b)  cube(i, j, k, b  )
      }
    }
}
beginxyz
```

[탐구문제] 지하 -64층의 돌마당에서 시작하는 백층짜리 타워를 만들어봅시다.

```
beginxyz
If(   ?   ) return 1
(백층짜리 타워 만드는 코드)

if(-2<z && z<2) return 0
```

if(z==-63) return 1 (1 은 풀밭이다)
하면 -64층 돌마당 위에 풀밭이 만들어짐

<= 이 명령이 없으면, 자동으로 -1층에 돌이, 0층에 풀밭이, 일층 풀밭 끝에 담장이 만들어진다.

(참고) beginxyz 맨 끝에는, 쓰여있지는 않지만, 지하 일층과 0층 1층에 대한 명령이 있다.

다음은 중학교 이차함수의 좌표식을 활용하여 피요르드 해안을 만드는 아이디어입니다.

```
for( n=1; n<5 ; n++) {
    if( y<0.01*n*x*x +3*n -10  &&  z==6-n ) return n
}
```

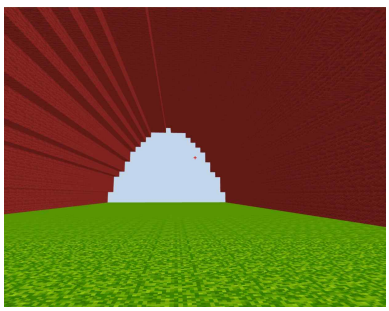
n=1	if(y<0.01*1*x*x + 3*1 -10 && z==6-1) return 1	y가 $0.01x^2$ - 7 보다 작고 5층(z==5)인 영역에 1번 벽돌로 만들어라.
n=2	if(y<0.01*2*x*x + 3*2 -10 && z==6-2) return 2	y가 $0.02x^2$ - 4 보다 작고 4층(z==4)인 영역에 2번 벽돌로 만들어라.
n=3		
n=4		

그렇다면 기본 식에서 제일 밑에 깔린 벽돌은 몇 번일까요? _____
터틀 코딩을 실행시키면 그 벽돌은 무슨 색인가요? _____

3) 코드 분석 (이차함수의 그래프을 이용하여 만들어본다)

아래의 코드를 한번 분석해 보겠습니다. ($y > -0.1x^2 + 20$ 을 생각하세요)

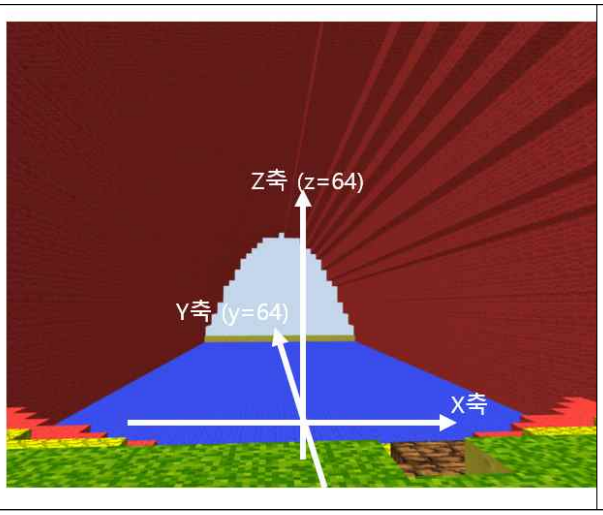
```
if (z > -0.1*x*x + 20) return 4
```

앞	뒤
	

왜 이런 터널 모양이 나왔을까요?

이유 ?

이 그림에서 z축과 x축을 찾을 수 있나요? 그림에 표시해봅시다.

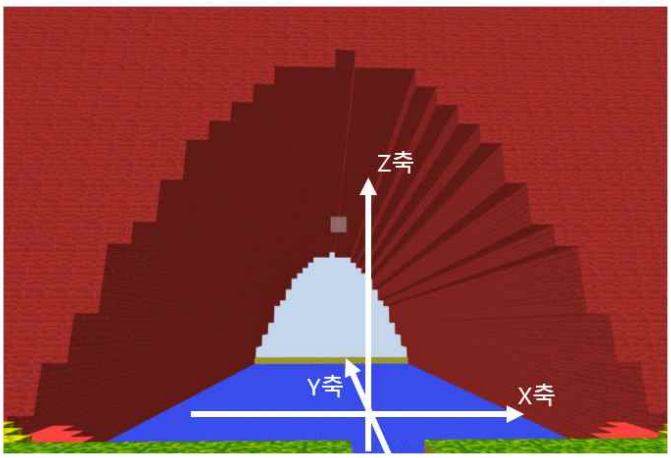



```
if (z > -0.1*x*x + 20)
return 4
```

위의 코딩식에서 y 범위를 지정해 주지 않았기 때문에 전체 y에 대해서 이차함수를 그린 것입니다.

그렇다면 y의 범위를 지정해 봅시다. (아래 명령은 틀림 : -5<y && y<5 이다)

```
if (z > -0.1*x*x + 20 && -5<y && y>5) return 4
```

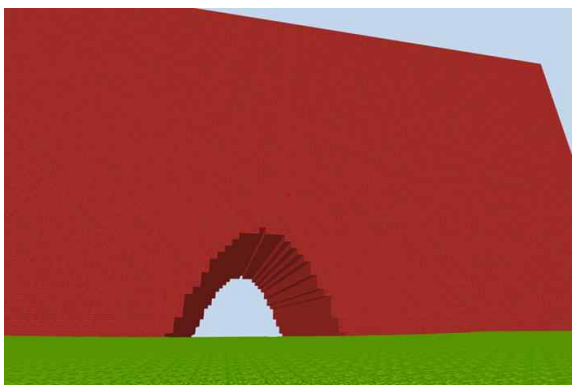


y의 범위를 주자 -5부터 5까지 터널의 길이가 짧아진 것을 볼 수 있습니다.

※※※ 주의!!!

범위를 줄 때는 -5<y<5 이렇게 주면 안 되고 -5<y && y<5 로 쪼개서 주어야 합니다. 터틀크래프트 뿐만 아니라 다른 컴퓨터 언어에서도 -5<y<5 명령을 쓰지 못하고 쪼개서 적어 주어야 하는 사실을 기억하자

좀 더 뒤로 가볼까요? 왜 이런 모양이 나왔다고 생각하나요?



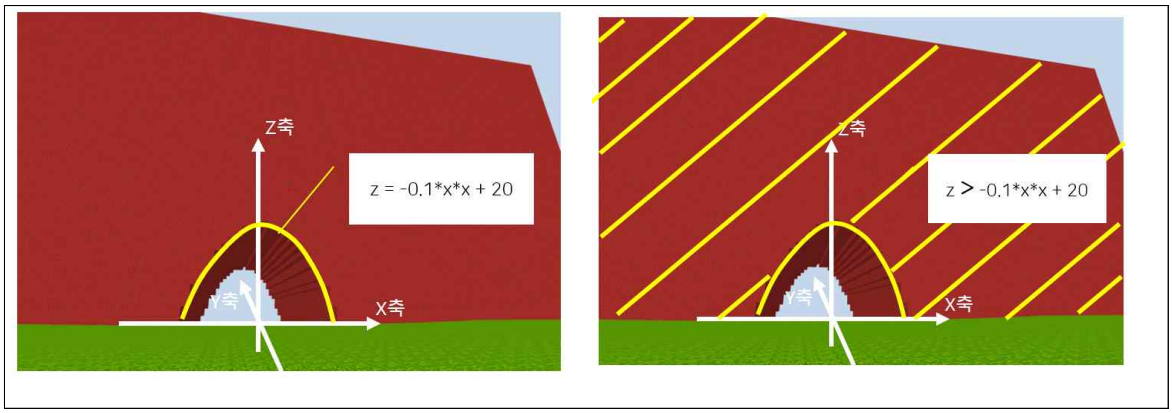
참고 2) 마인크래프트와 같은 게임에서는 w는 앞으로, a와 d 는 각각 왼쪽 오른쪽 그리고 s 는 뒤로 갑니다. 터틀크래프트에서는 x는 뛰면서 앞으로, z와 c 는 위와 아래로, sfift 는 중력을 작용시켜 거북이를 땅으로 !!!

다음 좌표 조건식을 살펴봅시다. 컴퓨터 코딩에서는 수학의 표현과 약간 다릅니다.

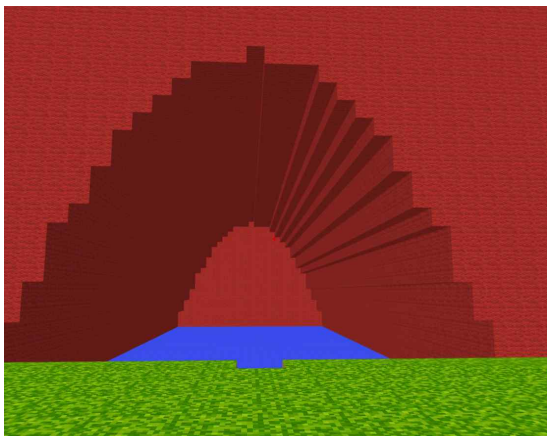
(참고) 아래는 $z > -0.1x^2 + 20$ 이면서 $-5 < y < 5$ 인 경우에 리턴 4

```
if (z > -0.1*x*x + 20 && -5<y && y<5) return 4
```

x로 주어진 이차함수 영역에서 이차함수보다 큰 z에 쌓기나무를 놓으라고 명령했기 때문입니다. 즉 처음에 z축과 x축만 보면 z축과 x축에 대한 이차함수를 발견할 수 있습니다.



자 이제 그러면 아래 그림처럼 한 쪽 면을 막아봅시다. 제일 끝에 부분의 y좌표가 y=64 라고 했죠? 코딩식을 한번 써보고 컴퓨터로 확인해봅시다.

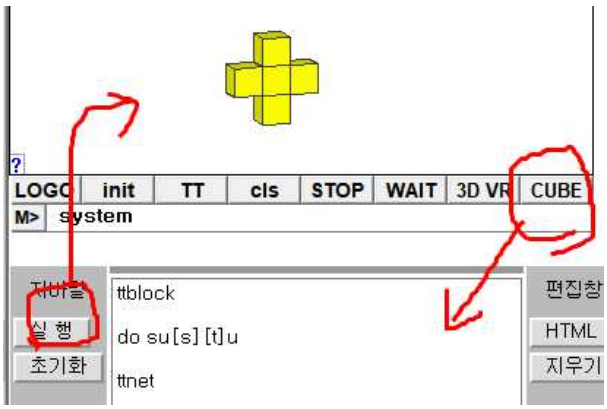


손으로 써봅시다

참고 1) 0 번 벽돌은 빈 공간을 만들고, 1번은 풀밭, 2번은 나무 기둥의 색, 3번은 잎의 색, 4번은 死 빨강색, 5는 ocean 파란색, 6은 다이아몬드, 7은 벽돌, 8은 나무벽돌, 9는 돌, 10은 거북이, 11은 또 다른 땅, 12는 나무, 13은 나뭇잎, 14는 핑크, 15는 얼음, 16은 고대 건축물 벽돌, 17은 벽돌, 18은 앵그리버드, 19는 벽돌, 그리고 20은 숫자 0 이며, 21부터 30까지 각각 1부터 10까지의 숫자를 나타내는 블록이 만들어집니다.

터북이 크래프트와 터틀말을 활용한 나만의 object 만들고 심기

위의 [터틀말] 단추를 누르면 아래와 같은 화면이 나옵니다. 편집기에 아래와 같이 명령을 쓰고 터틀말의 [실행] 단추를 눌러 실행시키면 아래와 같이 십자가 모양이 그려집니다. 안되는 경우, 파이어폭스의 부가기능 옵션에서 [Java 6.25] 를 [항상 사용] 으로 합니다.



위의 편집기에서, ttblock 은 쌓기나무준비와 ㄱ 이 합쳐진 영어 명령이고. ㄴ 은 ttnet 입니다. do su[s][t]u 는 do (하자) 로 거북에게 앞으로 s, 위로 u, 위치방향 기억하고 [, 앞으로 s, 기억된 곳으로] , 기억과 뒤로 [t] , 위로 u 해서 만듭니다. 이제, [큐브식] 또는 [큐브셑] 이라 쓰인 단추를 누르면, 아래와 같이 쓰입니다 (왼쪽은 큐브식, 오른쪽 큐브셑).

X = 0 ; Y = 0 ; Z = 0 ; C = 7 ;
cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C);
cube(2+X, 1+Y, 2+Z, C);
cube(0+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C);
	function cubeset(X, Y, Z, C) {
	cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C);
	cube(2+X, 1+Y, 2+Z, C);
	cube(0+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C);
	} ; cubeset(0, 0, 0, 7)

여기에 나오는 cube 명령어는 x, y, z 좌표 뒤에 문자 X, Y, Z 이 쓰여져 있다.

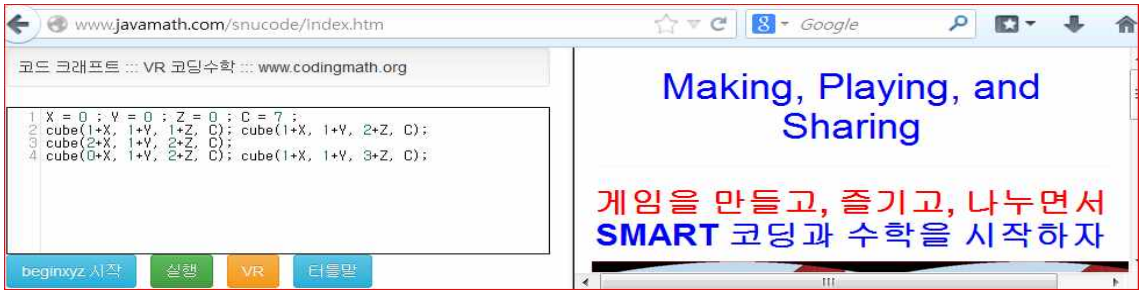
cube(1+X, 1+Y, 1+Z, C);	벽돌 한 개를 (1+X, 1+Y, 1+Z)좌표에 n번 블록으로 만들어라. (왜냐하면 위에서 C=n 으로 정의해놓았기 때문입니다.)

X=0, Y=0, Z=0 이기에, 대문자 X, Y, Z 가 처음에 모두 0 이다. color C 는 7 이다. 따라서, 대문자 X, Y, Z, C 의 값을 대입하여 변수 문자를 없애면 아래와 같은 명령이 된다.

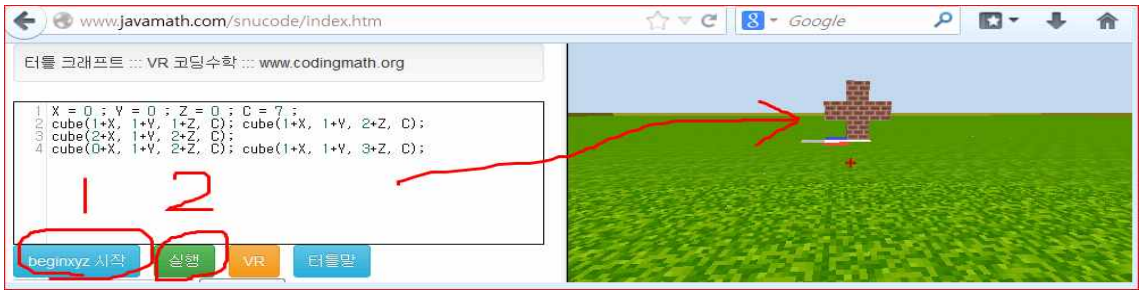
cube(1, 1, 1, 7); cube(1, 1, 2, 7); cube(2, 1, 2, 7);
cube(0, 1, 2, 7); cube(1, 1, 3, 7);

이 cube 큐브식은 7번 벽돌로 십자가를 이루는 다섯 개의 쌓기나무 cube를 만듭니다.

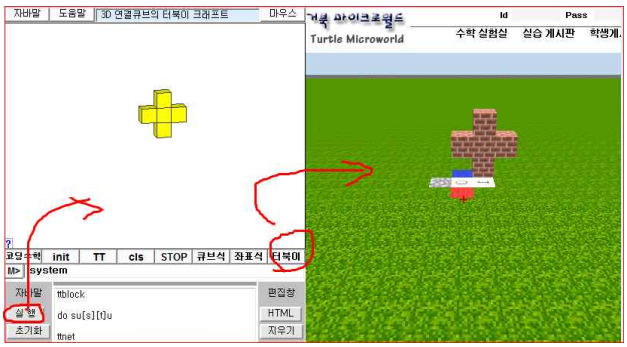
이제 편집기에 쓰인 큐브식을 아래와 같이 터틀 크래프트 편집창을 모두 청소한 후, 아래와 같이 그 곳에 터틀말에서 얻은 큐브식을 복사하고 실행시켜 봅니다.



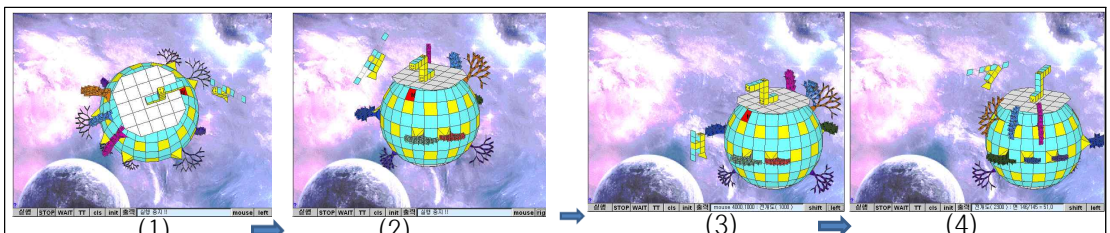
이제, 먼저 [beginxyz 시작] 단추로 배경 화면이 나오게 합니다. 그 다음, [실행] 단추를 누르면 편집기에 쓰인 큐브식이 실행되어 십자가 모습이 크래프트 화면에 나타납니다.



앞에서 다룬 **beginxyz 밑에 쓰인 좌표식은 [beginxyz 시작] 단추로 실행이 됩니다.** 그렇지만, **[큐브식] 단추로 만든 명령은 beginxyz 위에 쓰고 [실행] 단추를 눌러 실행시킵니다.** 아래 화면과 같이 터틀말에서 [터북이] 단추를 누르면 터틀 크래프트에 그려지기도 합니다.



** 터틀말에 그려진 십자가 모양을 마우스로 회전시키려면 아래의 사용법을 참고하세 **

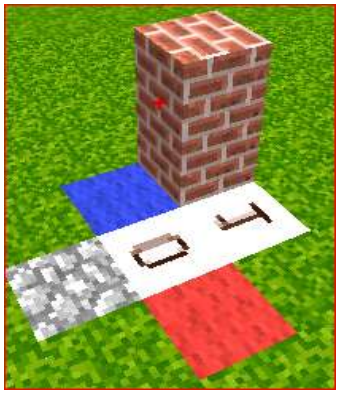


(1) 명령 [실행] 후, [STOP] 단추로 중지	(3) Shift+left 마우스 버튼 끌기로 이동
(2) 오른쪽 마우스 버튼을 끌어 회전	(4) Shift+right 마우스 버튼 애니메이션

[질문] 다음 두 명령은 같은 것을 그립니다 (오른쪽은 왼쪽 명령을 함수와 변수로 만든 것) 오른쪽과 같은 cubeset 함수 표현을 사용할 때 편리한 점이 무엇일까 ?

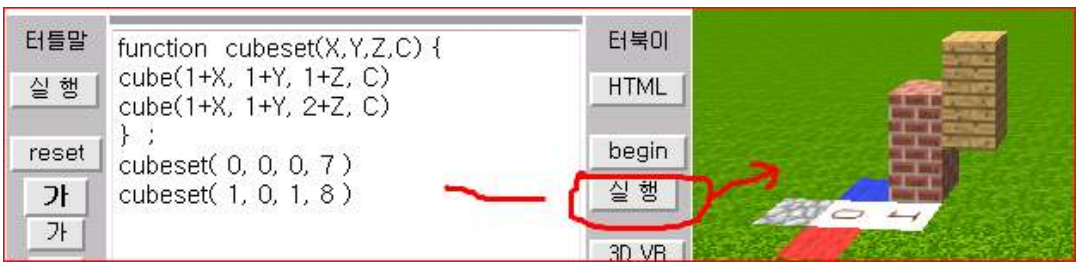
```
X=0; Y=0; Z=0; C=7
cube(1+X, 1+Y, 1+Z, C);
cube(1+X, 1+Y, 2+Z, C);
```

```
function  cubeset(X,Y,Z,C) {
cube(1+X, 1+Y, 1+Z, C)
cube(1+X, 1+Y, 2+Z, C)
}  ;   cubeset( 0, 0, 0, 7 )
```



[질문] 다음 왼쪽의 명령을 실행시킨 그림이 오른쪽과 같은 이유를 생각하고 설명해보자

cubeset(0,0,0,7) 과 cubeset(1, 0, 1, 8) 의 뜻은 무엇인가 ?



[질문] 아래 두 명령은 같은 결과를 만든다. 그 이유를 알아보자.

```
for( n=1; n<10; n++ ) {
X=n ; Y=0 ; Z=0 ; C=n
cube(1+X, 1+Y, 1+Z, C);
cube(1+X, 1+Y, 2+Z, C);
}
```

```
function  cubeset(X,Y,Z,C) {
cube(1+X, 1+Y, 1+Z, C)
cube(1+X, 1+Y, 2+Z, C)
}  ;
for( n=1; n<10; n++ ) {
   cubeset( n, 0, 0, n ) }
```

** for(n=1; n<10; n++) { cubeset(n, 0, 0, n) } 명령의 설명

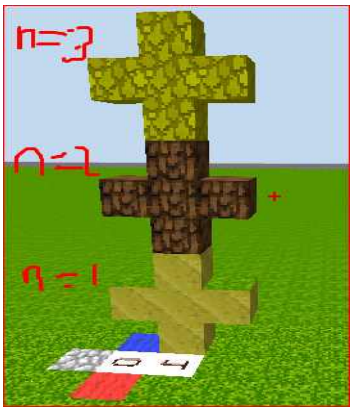
n=1 즉 변수 n에 1이 초기값으로 대입되고, n++ 즉 n=n+1 즉 n의 값이 하나씩 증가하면서 n<10 즉 n이 10 보다 작을 때 까지 { 과 } 안의 cubeset(n,0,0,n)을 실행한다. 결과적으로 cubeset(1,0,0,1) ; cubeset(2,0,0,2) ..... cubeset(9,0,0,9) ; 명령이 차례로 실행되어, 결과적으로 다음과 같은 블록의 모양이 만들어진다.

[실습] 앞에서 만든 십자가를 쌓아 올리는 다음 2가지 명령을 설명하고 실습하여보자.

```
for( n=0; n<3; n++) {
X = 0 ; Y = 0 ; Z = 3*n ; C = n ;
cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C);
cube(2+X, 1+Y, 2+Z, C);
cube(0+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C);
}
```

```
function cubeset(X, Y, Z, C) {
cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C);
cube(2+X, 1+Y, 2+Z, C);
cube(0+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C);
} ;
for( n=0; n<3; n++) {
    cubeset(0, 0, 3*n, n+1)    }
```

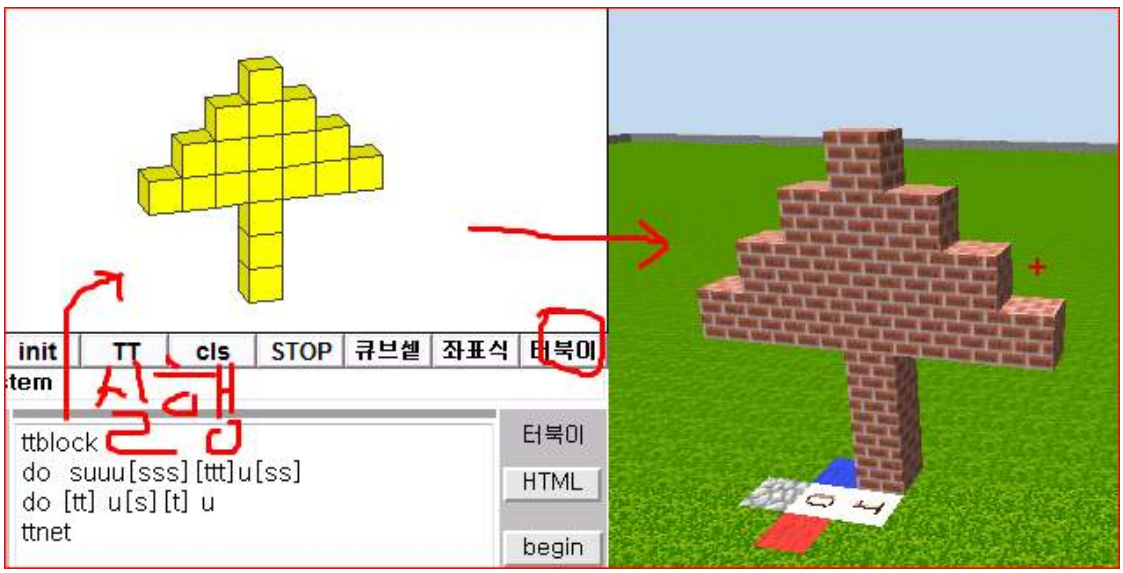
두 명령은 같은 결과를 만든다. 두 가지 명령의 장점과 단점을 각각 비교하여 생각해보자.



for(n=0; n<3; n++) { cubeset(0, 0, 3*n, n+1) } 명령은 n=0에서 시작해서, n++ 즉 n의 값이 하나씩 증가하며 n<3 즉 n이 0, 1, 2 일 때까지 { 과 } 안의 cubeset(0, 0, 3*n, n+1) 이 실행된다. 따라서, n=0, n=1, n=2 인 경우에 각각 cubeset(0, 0, 0, 1) 과 cubeset(0, 0, 3, 2) 과 cubeset(0, 0, 6, 3) 이 차례로 실행이 된다,

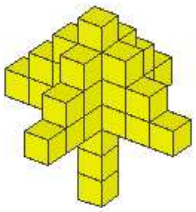
(참고) X, Y, Z, C 의 값에 따라 cube(1+X, 1+Y, 1+Z, C) 의 시작점과 벽돌의 모양이 달라집니다. 따라서 cube(1+X, 1+Y, 1+Z, C) 는 중학교에서 배우는 함수 개념을 빌리면, 변수 X, Y, Z, C 에 따라 변하는 함수입니다. (고등학교 대응개념으로의 함수)

십자가를 넘어, 나무의 모양을 만들어보자



그런데, [터틀말] 사용법은 무엇인가 ? 간단한 설명이 앞 부분에 있다 !!

본격적으로 네 방향으로 나무 모양이 나게 하려면 다음과 같은 거북 코드가 필요하다.



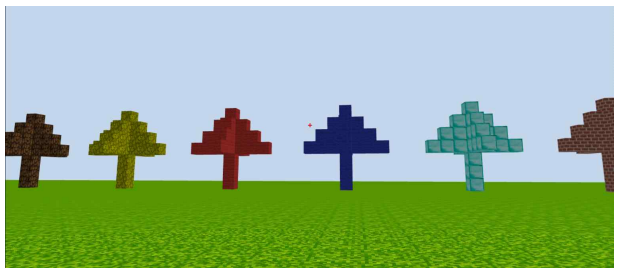
init	TT	cls	STOP	WAIT
lem				

```
ttblock
do suuu[sss][Lsss][Rsss][ttt]
do u[ss][Lss][Rss][tt]
do u[s][[Ls][Rs][t]u
ttnet
```

ttblock
do suuu[sss][Lsss][Rsss][ttt]
do u[ss][Lss][Rss][tt]
do u[s][[Ls][Rs][t]u
ttnet
(s는 앞으로, t는 뒤로, L R left right, [와])

이제 위 코드로 네 방향으로 나가는 아래와 같은 나무 cube 큐브식을 얻자. beginxyz 로 큰 세계관을 만들었으면 그 안에 여러 요소(object)들을 배치해봅시다. 여기서는 일단 주어진 기본 코드를 통해 나만의 나무를 만들어 심어보는 연습을 하겠습니다. 위에서 배경을 만들 때는 beginxyz를 눌렀다면 object들을 만드는 것은 추가하기에 **'[실행]'** 버튼을 누릅니다.

```
for ( n=1; n<10; n++ ) { X=50-10+n; Y=-40; Z=5; C=n;
    cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C); cube(1+X, 1+Y, 4+Z, C);
    cube(2+X, 1+Y, 4+Z, C); cube(3+X, 1+Y, 4+Z, C); cube(4+X, 1+Y, 4+Z, C); cube(1+X, 2+Y, 4+Z, C);
    cube(1+X, 3+Y, 4+Z, C); cube(1+X, 4+Y, 4+Z, C); cube(1+X, 0+Y, 4+Z, C); cube(1+X, -1+Y, 4+Z, C);
    cube(1+X, -2+Y, 4+Z, C); cube(0+X, 1+Y, 4+Z, C); cube(-1+X, 1+Y, 4+Z, C); cube(-2+X, 1+Y, 4+Z, C);
    cube(1+X, 1+Y, 5+Z, C); cube(2+X, 1+Y, 5+Z, C); cube(3+X, 1+Y, 5+Z, C); cube(1+X, 2+Y, 5+Z, C);
    cube(1+X, 3+Y, 5+Z, C); cube(1+X, 0+Y, 5+Z, C); cube(1+X, -1+Y, 5+Z, C); cube(0+X, 1+Y, 5+Z, C);
    cube(-1+X, 1+Y, 5+Z, C); cube(1+X, 1+Y, 6+Z, C); cube(2+X, 1+Y, 6+Z, C); cube(1+X, 2+Y, 6+Z, C);
    cube(1+X, 0+Y, 6+Z, C); cube(0+X, 1+Y, 6+Z, C); cube(1+X, 1+Y, 7+Z, C);
}       // [실행]으로 비긴xyz 워를 실행시켜 미로 만듦
```



터틀크래프트를 처음 시작하면 이런 식이 나오는데, 중괄호 안의 cube 명령는 무엇일까요?

cube(1+X, 1+Y, 1+Z, C);	벽돌 한 개를 (1+X, 1+Y, 1+Z)좌표에 n번 블록으로 만들어라. (왜냐하면 위에서 C=n 으로 정의해놓았기 때문입니다.)

박스 안에 있는 식은 나무 한 개만을 의미합니다. cube 명령으로 벽돌을 한 개씩 쌓아서 나무 한 개를 만들어놓은 상태입니다. 그런데 우리가 실행을 눌렀을 때는 여러 개의 나무가 심겨있죠? 어떤 명령어 때문일까요?

```
for ( n=1; n<10; n++ ) { X=50-10*n; Y=-40; Z=5; C=n;
   cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C); cube(1+X, 1+Y, 4+Z, C);
   cube(2+X, 1+Y, 4+Z, C); cube(3+X, 1+Y, 4+Z, C); cube(4+X, 1+Y, 4+Z, C); cube(1+X, 2+Y, 4+Z, C);
   cube(1+X, 3+Y, 4+Z, C); cube(1+X, 4+Y, 4+Z, C); cube(1+X, 0+Y, 4+Z, C); cube(1+X, -1+Y, 4+Z, C);
   cube(1+X, -2+Y, 4+Z, C); cube(0+X, 1+Y, 4+Z, C); cube(-1+X, 1+Y, 4+Z, C); cube(-2+X, 1+Y, 4+Z, C);
   cube(1+X, 1+Y, 5+Z, C); cube(2+X, 1+Y, 5+Z, C); cube(3+X, 1+Y, 5+Z, C); cube(1+X, 2+Y, 5+Z, C);
   cube(1+X, 3+Y, 5+Z, C); cube(1+X, 0+Y, 5+Z, C); cube(1+X, -1+Y, 5+Z, C); cube(0+X, 1+Y, 5+Z, C);
   cube(-1+X, 1+Y, 5+Z, C); cube(1+X, 1+Y, 6+Z, C); cube(2+X, 1+Y, 6+Z, C); cube(1+X, 2+Y, 6+Z, C);
   cube(1+X, 0+Y, 6+Z, C); cube(0+X, 1+Y, 6+Z, C); cube(1+X, 1+Y, 7+Z, C);
}       // [실행]으로 비긴xyz 위를 실행시켜 미로 만듦
```

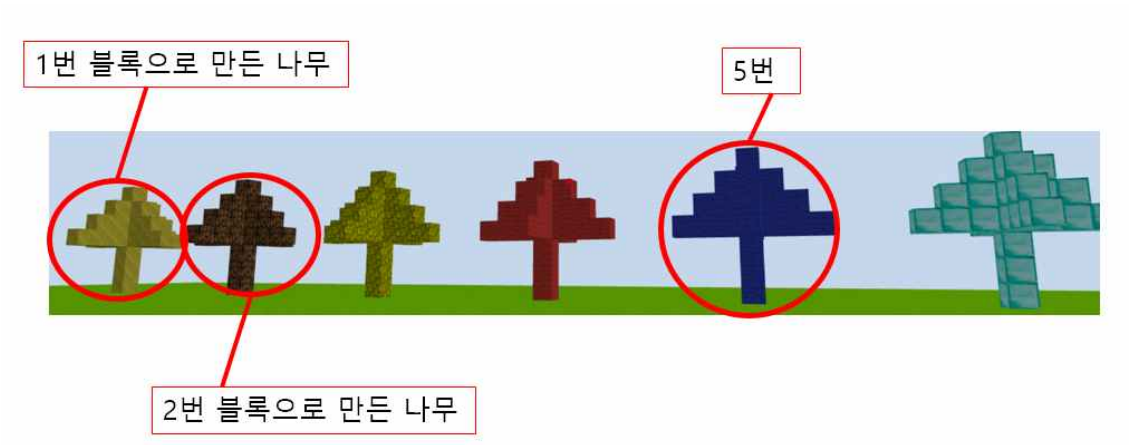
바로 제일 윗줄의 for 명령어 때문입니다. 앞에서 beginxyz를 하며 배운 명령어죠?

`for ( n=1; n<10; n++ ) { X=50-10*n; Y=-40; Z=5; C=n;`
n이 1일 때부터 9일 때까지 n이 하나씩 늘어나면서 { } 안의 명령을 수행 하여라.

빈 칸을 한 번 채워볼까요?

<table>
<tr><td colspan="3"><pre>
for (n=1; n<10; n++) { X=50-10*n; Y=-40; Z=5; C=n;
   cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C); cube(1+X, 1+Y, 4+Z, C);
   cube(2+X, 1+Y, 4+Z, C); cube(3+X, 1+Y, 4+Z, C); cube(4+X, 1+Y, 4+Z, C); cube(1+X, 2+Y, 4+Z, C);
   cube(1+X, 3+Y, 4+Z, C); cube(1+X, 4+Y, 4+Z, C); cube(1+X, 0+Y, 4+Z, C); cube(1+X, -1+Y, 4+Z, C);
   cube(1+X, -2+Y, 4+Z, C); cube(0+X, 1+Y, 4+Z, C); cube(-1+X, 1+Y, 4+Z, C); cube(-2+X, 1+Y, 4+Z, C);
   cube(1+X, 1+Y, 5+Z, C); cube(2+X, 1+Y, 5+Z, C); cube(3+X, 1+Y, 5+Z, C); cube(1+X, 2+Y, 5+Z, C);
   cube(1+X, 3+Y, 5+Z, C); cube(1+X, 0+Y, 5+Z, C); cube(1+X, -1+Y, 5+Z, C); cube(0+X, 1+Y, 5+Z, C);
   cube(-1+X, 1+Y, 5+Z, C); cube(1+X, 1+Y, 6+Z, C); cube(2+X, 1+Y, 6+Z, C); cube(1+X, 2+Y, 6+Z, C);
   cube(1+X, 0+Y, 6+Z, C); cube(0+X, 1+Y, 6+Z, C); cube(1+X, 1+Y, 7+Z, C);
}   // [실행]으로 비긴xyz 위를 실행시켜 미로 만듦
</pre></td></tr>
<tr><td>n=1</td><td>X=40, Y=−40,
Z=5, C=1</td><td>cube(41, −39, 6, 1),
cube(41,−39,7,2) ..... cube(41,−39,12,1)</td></tr>
<tr><td colspan="3">1번 블록으로 (41,-39,6), (41,-39,7) ..... (41,-39,12)에 하나씩 쌓아라.</td></tr>
<tr><td>n=2</td><td>X=30, Y=−40,
Z=5, C=2</td><td>cube(31, −39, 6, 2)
cube(31,−39,7,2)..... cube(31,−39,12,2)</td></tr>
<tr><td colspan="3">2번 블록으로 (31,-39,6), (31,-39,7) ... (31,-39,12) 에 하나씩 쌓아라.</td></tr>
<tr><td>n=3</td><td></td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>n=9</td><td></td><td></td></tr>
<tr><td colspan="3"></td></tr>
</table>

실행 결과를 보면 줄 세워져 있는 모습을 볼 수 있습니다.



[탐구문제] 컴퓨팅 사고력의 핵심적인 분야는 추상화와 자동화입니다 (Abstraction 과 Automation). 추상화와 자동화를 익혀봅니다

원하는 모양을 만들고 여러 개를 한 번에 심어봅시다. 원하는 모양을 추상적인 어떤 명령어로 만들고, 어떻게 변수 X Y Z C를 변화시키며 자동화 시켜서 여러 개를 심게 해줄까요? 추상화와 자동화로 그동안 배운 것들을 통합하여 익히자. 자동화에는 for (n=1 ; n<10 ; n++) { } 와 같이 n=1부터 10 미만까지, n이 하나씩 커지며 { } 안의 명령이 실행되게 하는 for 반복명령이 유용하게 쓰입니다. 이제 다음 큐브 명령어를 탐구해보자.

```
cube(1+X, 1+Y, 1+Z, C); cube(1+X, 1+Y, 2+Z, C); cube(2+X, 1+Y, 2+Z, C); cube(0+X, 1+Y, 2+Z, C); cube(1+X, 1+Y, 3+Z, C);
```

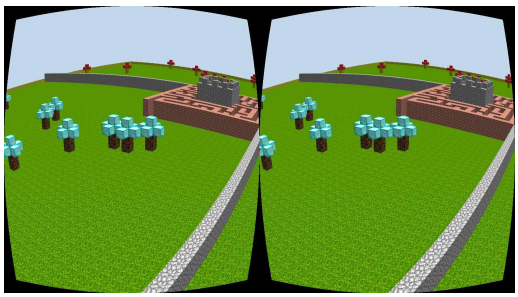
(참고 !!!) 아래의 명령어에서 ? 는 여러분들이 만들고 싶은 오브젝트의 개수를 나타낸다. 1,2,3, 은 각각 그것들을 어떤 모양으로 배열할 것인가를 고민하게 하는 부분입니다. ? 1 2 3 을 고치며 탐구해보자.

```
for(n=1; n< ? ; n++) { X= 1 ; Y= 2 ; Z= 3 ;}
```

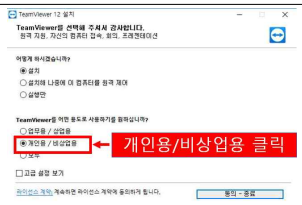
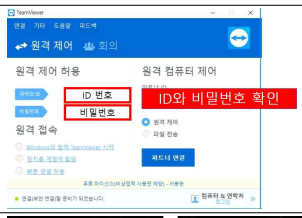
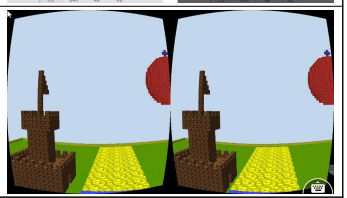
터틀크래프트로 만든 작품을 VR 연결하여 보는 방법

작품을 VR로 감상하려면, 터틀 크래프트 아래의 'VR' 단추를 클릭하면 된다.

⇒

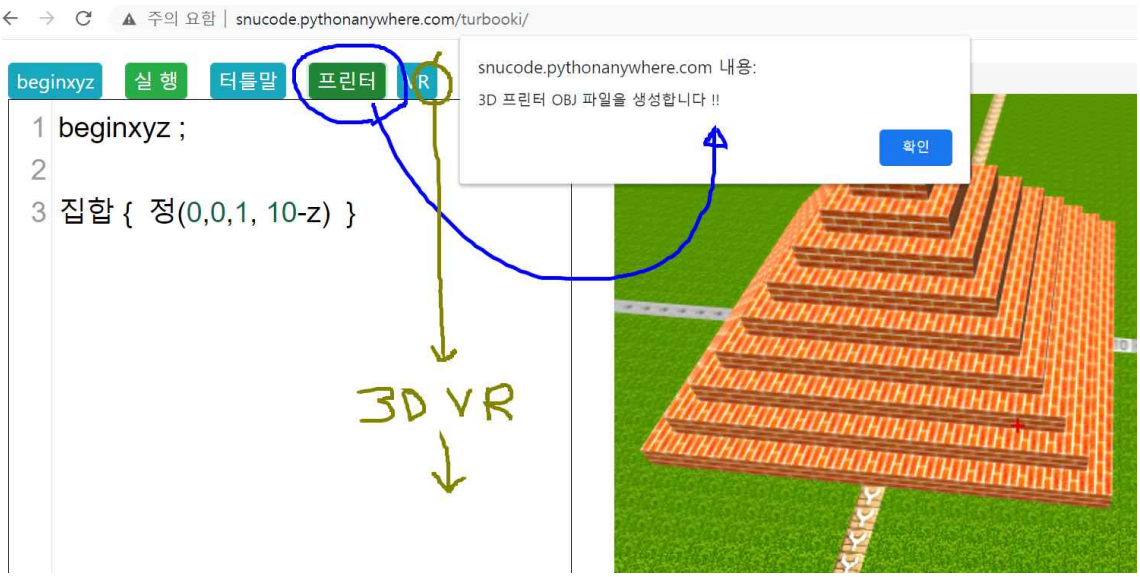


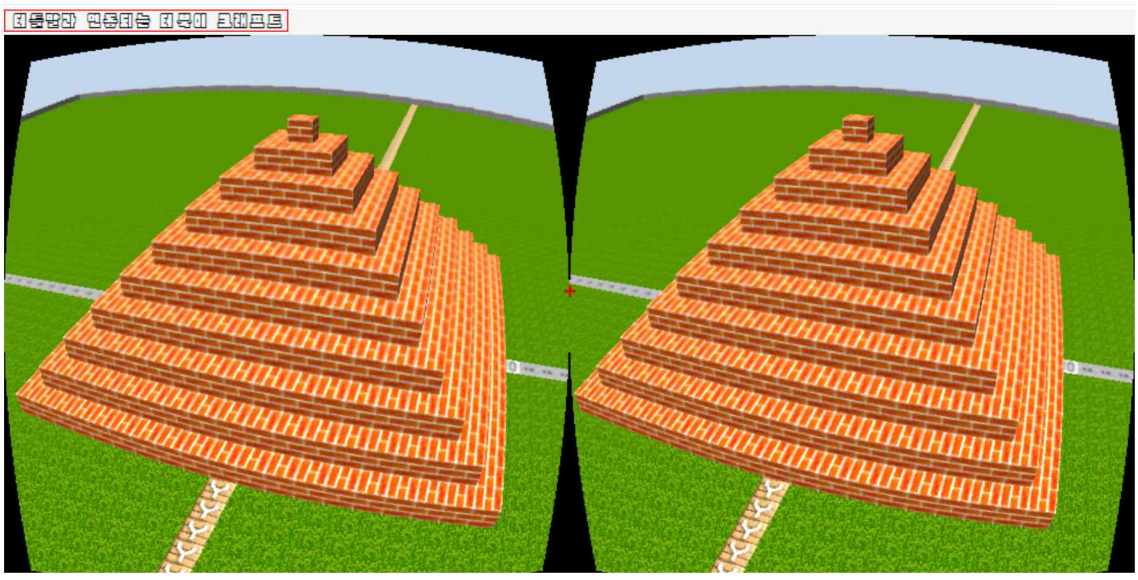
새롭게 만들어진 VR창에는 실행 결과가 양쪽으로 분할되어 보인다. 이를 다음과 같은 과정을 통해 스마트폰을 이용한 VR로 감상해 보자.

Step❶	PC원격 제어 프로그램인 '팀 뷰어'를 이용하여, 컴퓨터VR 화면을 스마트폰으로 옮기자. 스마트폰에서 '팀 뷰어' 어플리케이션을 검색하여 다운로드 받는다.	
Step❷	컴퓨터 인터넷 검색창에 '팀뷰어'를 검색하고, 프로그램을 다운받는다.(https://www.teamviewer.com/)	
Step❸	오른쪽과 같이 '개인용/비상업용'을 선택하고, '동의-종료'를 클릭하여 설치한다.	
Step❹	컴퓨터에 설치된 '팀뷰어' 프로그램을 실행하여. 아이디와 비밀번호를 확인한다.	
Step❺	스마트폰의 '팀뷰어' 어플을 실행하여 아이디와 비밀번호를 입력한다.	
Step❻	스마트폰으로 화면이 연결된 것을 확인하고, VR 기기에 스마트폰을 넣는다. VR 게임을 플레이 한다.	

터틀크래프트 코딩작품을 3D 프린터로 뽑아내는 방법
snucode.pythonanywhere.com/turbooki (임시로 hanaone.com)

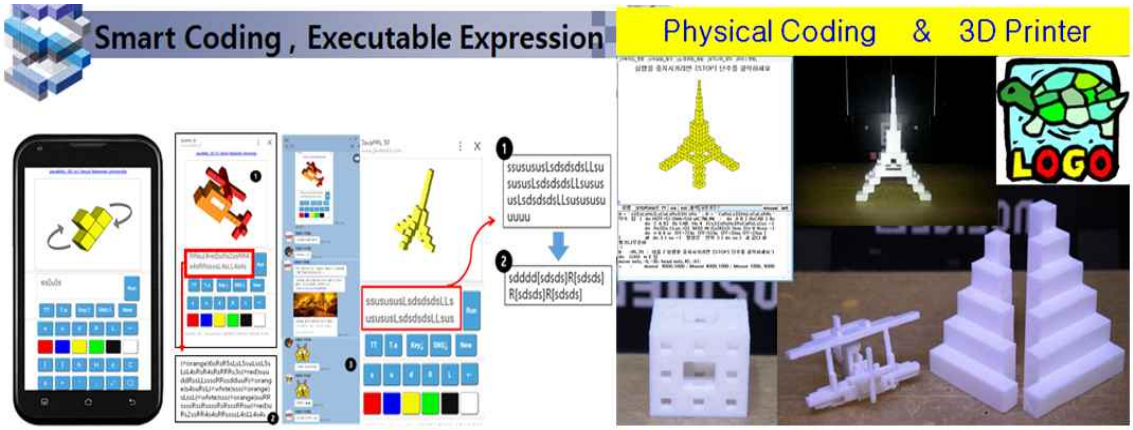
[3D 프린터와 VR 사용법] 위 주소에서 아래와 같이 코딩수학으로 입체를 만들고, 그 것을 3D 프린터와 VR 로 놀아보는 방법을 알아보자.





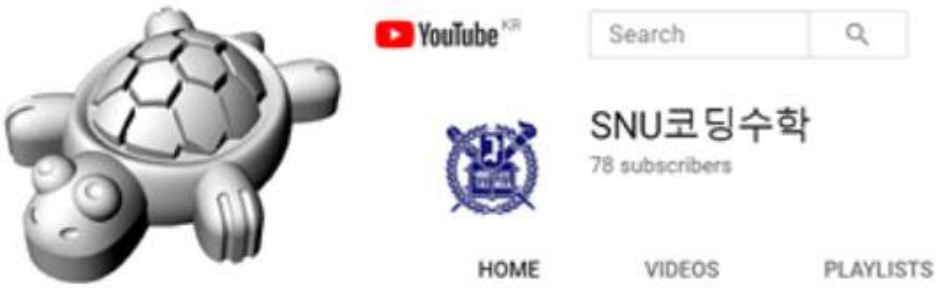
위와 같이 코딩수학으로 입체를 만드는 방법과 3D 프린터와 VR 등에 대한 내용은 우선 서울대학교 과학영재센터의 수리정보 분과 밴드와 유투브 동영상을 통해서 설명을 얻을 수 있습니다. (참고 내용 및 동영상 설명) https://band.us/band/66671454/post/88

더 자세하게 탐구하려면, 다음에 제시되는 codingmath.org 와 밴드 유튜브를 참고하세요 !!



codingmath.org 와 band.us/@snucode

아래는 서울대 코딩수학 탐구 및 snucode.org에서 진행되고 있는 서울대와 시흥시가 함께 하는 창의코딩 및 코딩수학 유튜브 동영상 주소입니다.



3D 프린터 메뉴와 마우스 사용법

([코딩수학]_창의 = [어린왕자준비~잎] 클릭) 거북명령 화면의 [?] 표시를 클릭하면 거북 실행화면이 나타나고, 그 화면에서 [어린왕자준비~잎]을 선택하여 클릭한다.

(1) 오른쪽과 같이 **[코딩수학]_창의** 메뉴에서 **[어린왕자준비~잎]**을 선택하면, 아래 부분의 편집기에 명령이 쓰이고 실행 화면이 나타난다.

(2) 예를 들어, 편집기의 맨 위에 쓰여 있는 **약속 땅 { do L S u ... }** 부분이 어린왕자 별의 북극에 위치하고 있는 평평한 땅에 쌓기나무로 무엇을 만드는 명령이 있는 곳이다. 이 부분을 고친 후, [실행] 단추를 눌러 실행시켜 보자.

(3) 그 아래의 **약속 우 { do 명령 ...}** 부분이 태양 전지판이 도는 우주선을 쌓기나무로 만드는 곳이다. 여기도 고쳐서 멋있게 만들어보자.

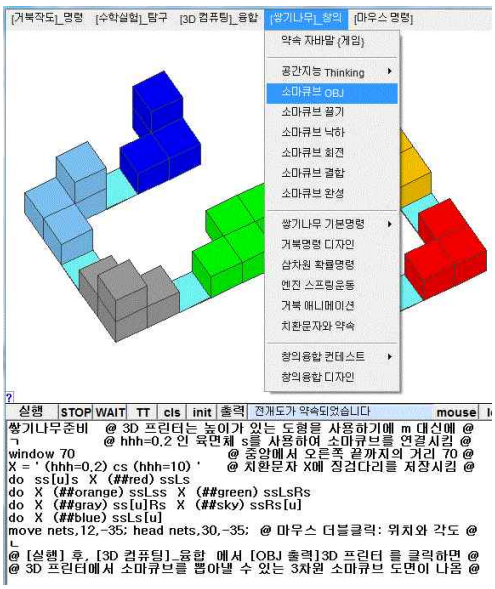
(4) 명령 편집기에 쓰인 쌓기나무 거북명령을 고치고 실행시키며 창의융합 작품을 만든 후, 마우스로 명령을 복사한 후 게시판에 이를 저장한다. 게시판에서 글쓰기로 들어가 게시판 편집기에서 직접 작품을 만들며 실행하면 편하다.

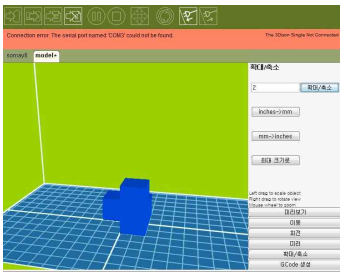
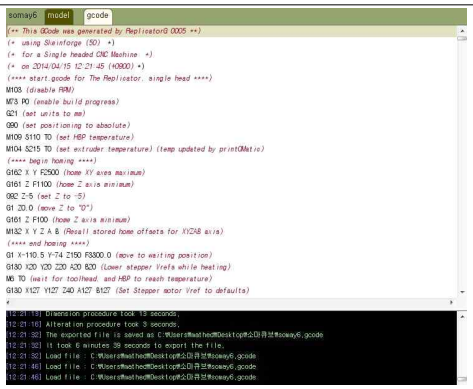
(1) **[코딩수학]_창의** 에서 [**공간지능 Thinking**]을 선택하고, [**소마큐브 3D 프린터**] 선택하면 화면과 같이 소마큐브 7 조각이 나타난다. 이 때, 편집기에 그림을 만드는 거북명령이 쓰인다.

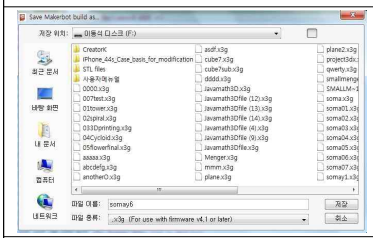
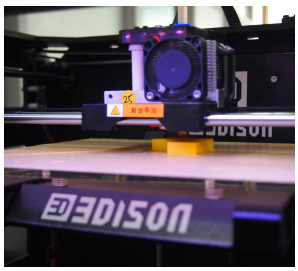
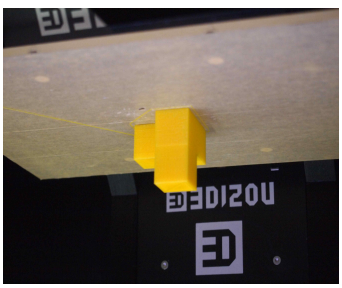
(2) 메뉴판에서, **[3D 컴퓨팅]_융합** 메뉴를 선택하고 제일 먼저 나타나는 **[3D 프린터] OBJ 파일**을 선택하면 화면의 소마큐브를 3D 프린터에서 뽑아내는데 사용되는 OBJ 정보가 편집기에 쓰여진다. (이 때, v 는 점들의 좌표, f 는 면의 정보)

(3) 편집기 내용을 선택한 후, 메모장에 복사하고, 이 것을 test.obj 파일로 저장하고, 저장된 obj 파일을 3D 프린터에서 부른 후, 뽑아낸다.

(4) 소마큐브에 대한 여러 메뉴판의 명령을 통해, 버추얼 공간에서 소마큐브를 끌고, 결합시켜 소마큐브를 만들고 조작해 본다.



<table>
<tr>
<td>  </td>
<td>A. 프린터 출력전용 소프트웨어 (예: CreatorK) 를 열고 만들어진 obj일 파일을 화면 창에 드래그하여 예상 출력 모양이 화면 창에 뜨는 것을 확인해보아요. 여기서 Creator K는 재차 버전업되고 있으며 3D Edison 프린터와 호환하여 인쇄 효율성을 높이기 위한 제작사의 최적화된 소프트웨어랍니다.</td>
</tr>
<tr>
<td> 
</td>
<td> 

B. 여러 가지 출력 상황을 조정(크기, 출력위치, 채우기, 서포터 설정 등)한 후 gcode를 생성하는 버튼을 눌러보아요. gcode는 거북명령어를 3D프린터가 인식할 수 있는 언어랍니다.</td>
</tr>
<tr>
<td>  </td>
<td>C. 만들어진 gcode를 확인한 뒤 SD카드에 확장자 x3g로 저장해요.</td>
</tr>
<tr>
<td> 
 </td>
<td>D. SD카드를 3D 프린터에 삽입하고 출력버튼을 눌러보아요. 그리고 최종 완성물을 확인해보아요. 이어서 남은 소마큐브를 3D프린터로마저 만들어 공룡을 완성해 보아요.

 </td>
</tr>
</table>