2002 서울대학교 과학 영재 센터

정보분과
순환교육 교재 III.

차례

학습 목표

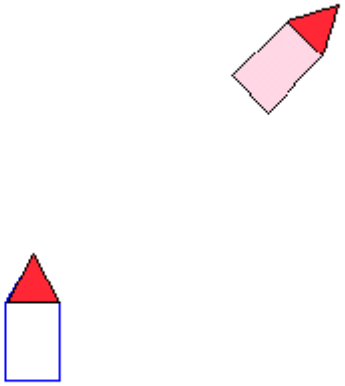
전 두 시간동안 배운 내용과 함께, 타일 명령을 기반으로 하여 학생들이 퍼즐 등의 작품을 만드는 open project를 실시한다.

게임 프로그래밍을 위해 **javamath.web.edunet4u.net** 에 있는 성냥개비 퍼즐, 소코반 게임 등을 활용하고, 이를 타일 명령으로 로고에서 구현한다.

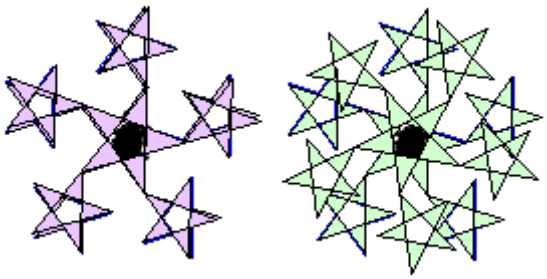
Ⅰ. 타일로고(TILE LOGO) 기본 명령

예전 만화 영화에서 그림만 그리면 그것이 실제로 살아 움직이곤 하는 일이 있었다. 우리가 잘 알고 있는 피노키오도 할아버지가 나무를 깎아서 만들어 놓은 인형에 불과했지만 스스로 움직이게 된다.

그것이 로고 그래픽을 이용해서도 가능하다.
로고 그래픽을 이용해 마음대로 그림을 그리고 그것을 원하는 대로 살아 움직이게 할 수 있으며 창의성을 발휘해서 게임을 만들 수도 있다.



<타일 로고를 이용한 로켓 발사>



<타일 로고를 이용해서 그린 '춤추는 별들'>

이번 시간에는 실제로 그렇게 만들어 보기 위해서 기본적으로 알아야 할 명령을 알아보자.

우선 그림을 움직이기 위해서는 그 단위를 어떻게 만들어야 할까?
로고 명령으로 그린 그림을 살펴보자. 로고 그래픽에서 기본적으로 그림에 나타나는 단위는 '선'이다. 이것을 '면' 이라는 단위로 바꾸어 움직이도록 할 것이다.

1. pt 명령

면을 만들기 위해서는 '꼭지점' 이라는 개념이 필요하다. 즉 점으로 말뚝을 박은 후, 이웃하는 말뚝들을 이어서 만들어지는 면이 하나의 타일을 구성하게 된다. 여기서 **pt 명령은 말뚝을 박는 명령**이다. 즉 우선 말뚝을 박을 위치로 거북이를 이동시킨 후 pt; 명령을 해준다. 예를 들어 사각형의 네 꼭지점에 말뚝을 박으려면

```
pt;
fd 30; rt 90; pt;
fd 30; rt 90; pt;
fd 30; rt 90; pt;
```

와 같이 해주면 말뚝이 박힌다. 실제로 말뚝은 보이지 않는다. 말뚝을 보고 싶으면 pt; 명령 다음에 point; 명령을 내려준다. 아래 그림은 point 명령을 이용해 말뚝이 보이도록 한 것이다.

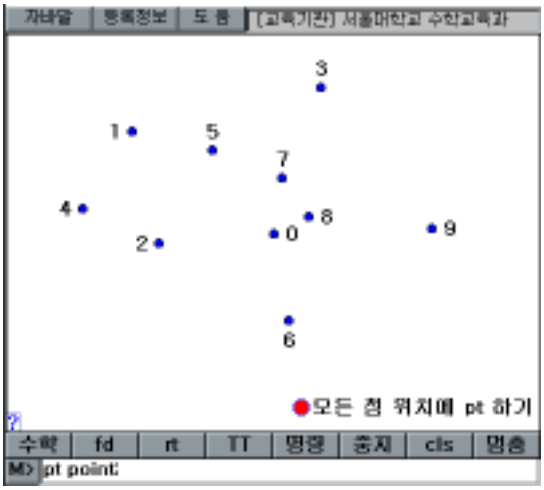
<사각형 타일을 만들기 위한 말뚝 박기>

이미 나타나 있는 점들을 말뚝으로 쓰려면 **pt point;** 명령을 해주면 된다. 아래 그림은 <수업 게시판>에서 pt 명령 연습을 위해 미리 점을 찍어 둔 그림이다. 참고로 그 명령은 다음과 같다.

```
CLS;
for i=0 to 9 ;
point v_i, i*10,5*i;
next;
```

실제로 이 상태에서 말뚝을 박기 위해 pt point;를 써주면 된다. 다른 방법으로 다음과 같이 쓰기도 한다.

```
for i=0 to 9;
pt v_i;
next;
```



<미리 저장된 점으로 말뚝 박기>

2. tile 명령

타일을 만들기 위해서는 박아놓은 말뚝을 이어주어야 한다. tile 명령을 이용해 말뚝을 이어 하나의 타일을 만들 수 있다.

(1) 색깔 바꾸기

tile을 만들 때 타일의 색을 마음대로 정해줄 수 있다. 색을 바꾸고 싶으면 tile 뒤에 색을 적어주면 된다. 예를 들어

tile yellow;

라고 쓰면 노란색 타일이 만들어지게 된다. 특히

tile random;

이라고 쓰면 자동적으로 색깔이 여러 가지로 지정된다. (random 은 '임의의' 라는 뜻을 가진 영어이다.)

(2) 이름 붙이기

타일에 이름을 붙이고 싶을 경우 tile 다음에 원하는 이름을 써준다. 예를 들면

tile A;

와 같은 방식이다.

※tile A, yellow - 타일 A를 처음 만드는 경우에는 노란색 타일을 만들게 되는 명령이고, 타일 A가 이전에 있었다면 A의 색을 노랑색으로 바꾸어주는 명령이다.
※타일의 이름을 알고 싶으면 타일에 마우스를 대고 더블 클릭한다. 그러면 위의 창에 이름이 나타난다.

(3) 새로 타일 만들기

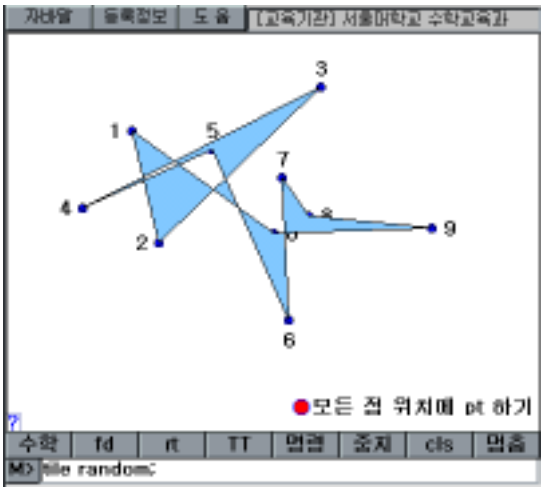
타일을 여러 개 만들 경우 새로운 타일을 만들기 전에 **new;** 명령을 해준다. 이 명령은 이전에 박은 말뚝을 지우는 명령이다. 즉 이전의 말뚝을 지우고 새로운 말뚝만 기억하게 된다.

(4) 타일 지우기

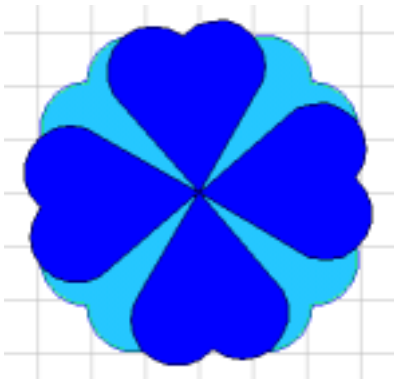
대문자로 **CLS;**를 하게 되면 타일과 거북이가 사라진다. 이전의 말뚝과 타일의 모양이 모두 사라지게 된다. 이때 거북이만 다시 보이게 하려면 **tt;** 명령을 한다.

(5) 타일 붙이기

여러 가지 타일을 만들어서 서로 붙이기를 하고 싶은 경우에는 타일의 이름을 나열해 주면된다. 예를 들어 타일 A와 B를 붙이고자 하는 경우에는 tile A,B 로 명령해 준다.



<임의의 점의 배치를 이용한 타일>



<타일을 이용한 네잎 클로버>

3. 마우스를 이용해 타일 움직이기

·옮기기

- 타일 위에 마우스 포인터를 놓고 클릭한 채, 원하는 방향으로 드래그해주면 타일이 자유롭게 움직인다.

·뒤집기

- 타일 위에 마우스 포인터를 놓고 마우스 오른쪽을 더블 클릭하면 타일은 뒤집어진 모양을 보여준다.

·돌리기

- 타일 위에 마우스 포인터를 놓고 오른쪽을 클릭한 채로 돌리고 싶은 방향으로 드래그하면 타일이 도는 모습을 볼 수 있다. 이 때 타일의 한 말뚝에 나타나는 동그라미가 회전의 중심이 된다.

·90도 회전

- shift를 누른 상태에서 타일을 클릭

·30도 회전

- ctrl을 누른 상태에서 타일을 클릭

4. 명령을 이용해 타일 움직이기

· tile A, fd, 숫자

- tile A를 정해진 수만큼 이동하게 하는 명령이다.
  이때 타일이 움직이는 방향은 거북이가 박은 첫 번째 말뚝을 기준으로 하여 두 번째 말뚝이 있는 방향을 향한다.

· tile A, rt, 숫자

- tile A를 정해진 각만큼 회전하게 하는 명령이다. 이때 타일이 회전하는 방향은 거북이가 박은 첫 번째 말뚝을 기준으로 하여 시계 반대 방향이다.

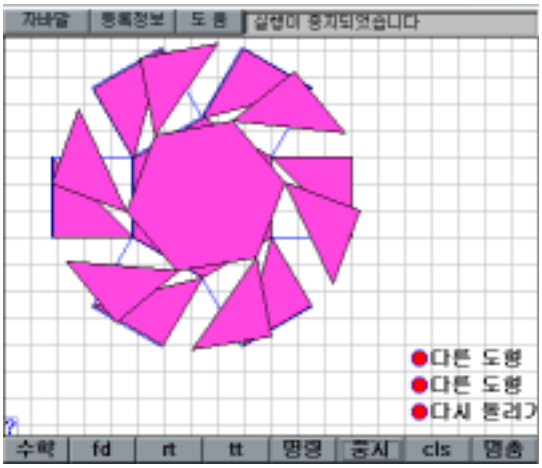
· axis 명령

- 그림판이 모눈 판으로 바뀌게 하는 명령이다. 뒤에 입력하는 숫자에 따라 눈금의 간격이 달라진다. 모눈을 따라 거북이의 위치를 지정해 줄 수 있다.
  원래의 그림판으로 돌아가려면 숫자 없이 axis를 입력해 준다.

· pause 명령

- 뒤에 오는 숫자만큼 거북이가 쉬도록 하는 명령이다. pause 명령을 사용하면 움직이는 모양을 좀더 뚜렷이 볼 수 있다.

지금까지 배운 함수 명령과 이 시간에 배운 타일 명령을 이용해 돌아가는 해바라기를 만들어보자.



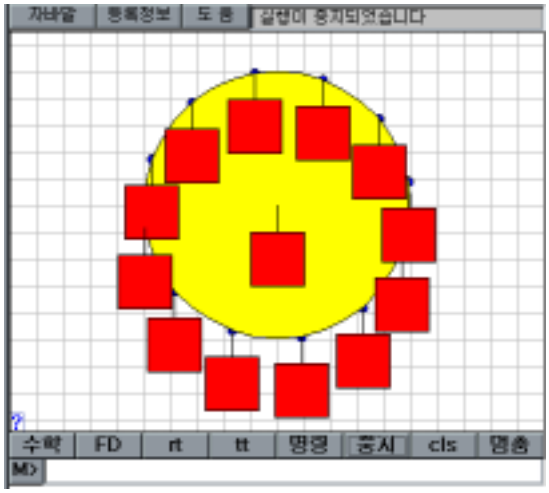
<타일 명령을 이용한 '돌아가는 해바라기'>

```
def f() {
CLS;tt;

반복 6 {
반복 4 { pt;가자 30;돌자 -90} 가자 30;돌자 60;}
tile random
new
tt -25,15;
pt;pt;pt;tile random;

tile t_1,t_0;
반복 36{tile t_1,rt,5;pause 10}
}

f();
```

Ⅱ. 허니문 카 와 성냥개비 게임

앞에서 배운 기본 명령을 바탕으로 두 가지 예제의 내용을 분석해 보자.

1. 허니문 카



<허니문 카>

```
CLS: tt;pu; tt 0,10,90;
pt;fd 50;pt;rt 95;
반복 36  {fd 314/36; rt 10 ; pt }
tile A, yellow;
tile A, 1;

new;tt 0,10,90;
pt;rt 180;fd 10;pt;
rt -90;fd 10;pt;
repeat 3  {rt 90;fd 20;pt;}
rt 90;fd 10;pt;
tile car,red;

for i=1 to 35 step 3  {point v_i,A,i ;}
point noname;

for i=1 to 35 step 3  {tile t_i,copy, car}

for i=1 to 35 step 3  {tile t_i,v_i;}

repeat 3600  {tile A, rt 5; pause 100}
```

주의: 컴퓨터가 이름을 정하여 점을 찍을 때 처음 것이 v_0 (v_1 이 아니다) 이다.
또한 pt 로 말뚝을 심을 때도 처음 것이 0 번째 말뚝이다.
마우스 왼쪽 버튼을 두 번 연속해 눌러 나타나는 점도 이름이 v_0부터
(v는 생략되고 뒤의 숫자만 화면에 나옴) 이다.

<명령어 분석>

> CLS: tt;pu; tt 0,10,90;

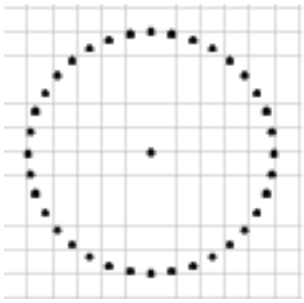
CLS 로 화면을 지운다. 이 때, 거북이 사라지므로 tt 로 나오게 한다.
pu (pen up)명령은 가자 명령에서 선을 그리지 않도록 한다 .
(선을 다시 그리게 하는 명령은 pd (pen down))
tt 0,10,90 은 좌표 (0,10) 에서 머리 각도가 90도로 향하게 한다.

> pt;fd 50;pt;rt 95;

첫번째 점을 찍고 (이 점이 회전이나 앞으로 갈 때의 기준점이 된다)
앞으로 50 가서 점찍고 95도를 돈다. 이 것은 좌표 (0,10) 을 중심으로 하고
반지름이 50 인 정 36각형을 그리려고 좌표 (0,10) 에서 50을 가고 95도 돈 것

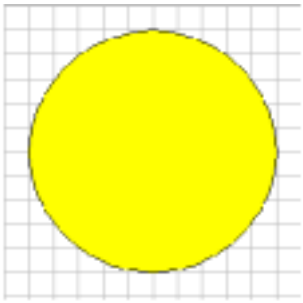
> 반복 36 { fd 314/36; rt 10 ; pt }

36각형을 그린다. 말뚝이 다음과 같이 박힌다.



> tile A, yellow;

36각형을 그리며 찍은 점 들을 연결한 A라는 이름의 타일로 만든다.



<허니문 카의 몸통>

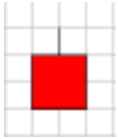
> tile A,
A 타일에서 1 번째 점부터 타일을 만든다.
default는 기준점인 0 번째 점부터 이지만 반지름에 해당하는 부분이 보이지 않도록 1번째 점부터 타일을 만든다. (기준점은 0 번째 점이다)

> new;tt 0,10,90;

새로 타일을 만들기 위해 new 명령

> pt;rt 180;fd 10;pt;
> rt -90;fd 10;pt;
> repeat 3 { rt 90;fd 20;pt;}
> rt 90;fd 10;pt;
> tile car,red;

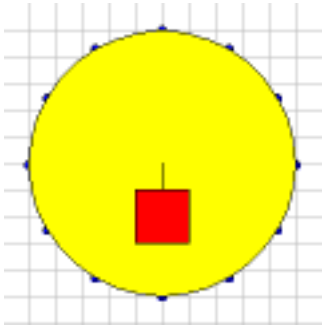
허니문 카에 달 승객용 방을 만든다.



<허니문 카의 승객용 방>

> for i=1 to 35 step 3 { point v_i,A,i };

i의 초기값이 1 부터 3씩 증가하면서 point v_i 를 타일 A의 i 번째 점에 붙인다. 여기서 pt 에 의한 점은 움직이지 않으나 point 에 의한 점은 움직이는 환경의 점이다. pt 와 point 에 의한 점은 다르다.



<허니문 카에 승객용 방 달기>

> point noname;

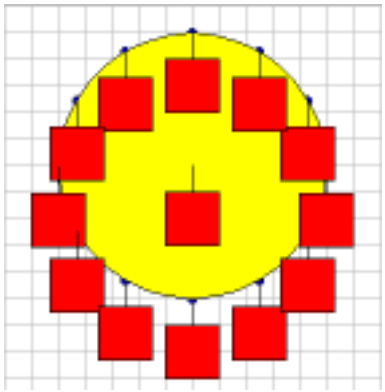
36각형 주변에 달린 점들의 이름이 안보이게 한다.

> for i=1 to 35 step 3 { tile t_i,copy, car }

앞에서 만든 승객용 차인 car를 복사하여t_i 라는 차량 타일을 여러 개를 만든다.

> for i=1 to 35 step 3 { tile t_i,v_i;}

차량 타일 t_i 를 점 v_i 에 단다.

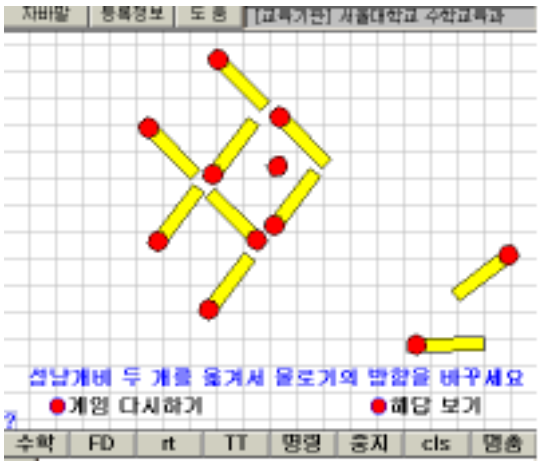


<허니문 카에 여러 개의 승객용 방 달기>

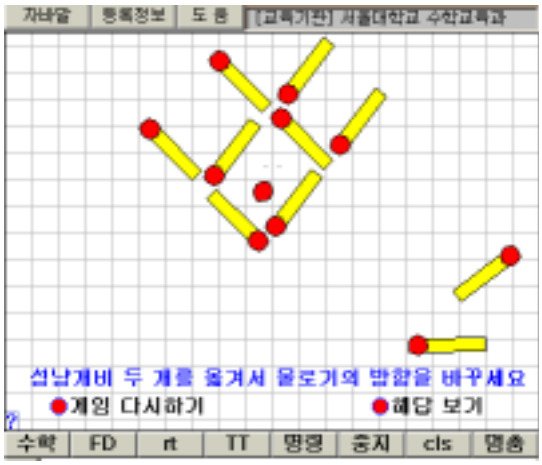
> repeat 3600 { tile A, rt 5; pause 100}

타일 A를 왼쪽으로 5 도씩 돌린다. 그리고100 ms 만큼 쉰다. 이 것을 3600번 반복한다. 타일 A가 돌때 회전축은 0 번째 점 즉 좌표 (0,10) 이다.

2. 성냥개비 게임



<성냥개비 게임>



<해답보기>

```
CLS; pu; axis 10;
pt;fd 2.5;pt;rt -90;fd 2.5;pt;
rt 90;fd 22.5;pt;rt 90;fd 5;pt;
rt 90;fd 22.5;pt;rt 90;fd 2.5;pt;
rt 90;pt;
for i=0 to 9;
```

```
tile yellow;
next;
new;fd 1.5;rt 100;repeat 18 {pt; fd 50/36; rt 20;}
tile A;tile A, red;

for i=0 to 9;
tile t_i, A;
next;

for i=0 to 9;
repeat 18 {tile t_i,fd 3; tile t_i,rt 3*i; pause 30;}
next;

def game() {
tile t_5, 54.5, -42.5, 360 ;
tile t_9, -5, -3.5, 135 ;
tile t_4, -45.5, 38.5, 315 ;
tile t_0, -23, -29.5, 53.13 ;
tile t_7, 3.5, 42.5, 315 ;
tile t_1, -19.5, 64, 315 ;
tile t_2, -42, -4, 53.13 ;
tile t_6, -21.5, 21, 53.13 ;
tile t_3, 1.5, 2, 53.13 ;
tile A, 2.5, 28, 206.57 ;
tile t_8, 89, -9, 216.87 ;
tt -98,-55;tt blue;
write '성냥개비 두 개를 옮겨서 물고기의 방향을
 바꾸세요';
}

button game() {'게임 다시하기'; -80,-65}
button sol() {'해답 보기'; 40,-65}
def sol() {
tile t_1, -19.5, 64, 315 ;
tile t_3, 1.5, 2, 53.13 ;
tile t_4, -45.5, 38.5, 315 ;
tile t_5, 54.5, -42.5, 360 ;
tile t_6, -21.5, 21, 53.13 ;
tile t_7, 3.5, 42.5, 315 ;
tile t_8, 89, -9, 216.87 ;
tile t_9, -5, -3.5, 135 ;
tile t_2, 6, 51.5, 53.13 ;
tile t_0, 25.5, 32.5, 53.13 ;
tile A, -3.5, 19, 206.57 ;
```

```
}

game();
```

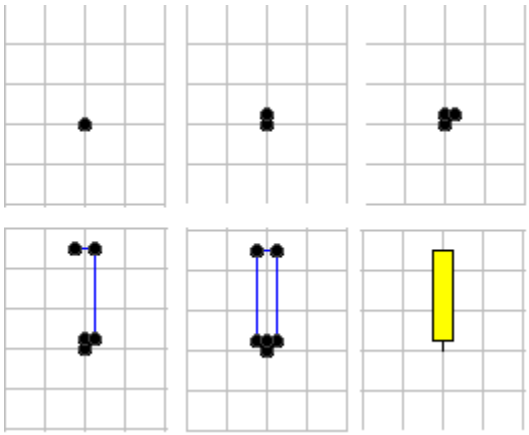
<명령어 분석>

```
> CLS; pu; axis 10;
```

CLS 로 화면을 지운다.
pu (pen up) 명령은 가자 명령에서 선을 그리지 않도록 한다 (선을 다시 그리게 하는 명령은 pd (pen down).
axis 10 으로 간격이 10인 모눈 판을 만든다.

```
> pt;fd 2.5;pt;rt -90;fd 2.5;pt;
> rt 90;fd 22.5;pt;rt 90;fd 5;pt;
> rt 90;fd 22.5;pt;rt 90;fd 2.5;pt;
> rt 90;pt;
> for i=1 to 10;
> tile yellow;
> next;
```

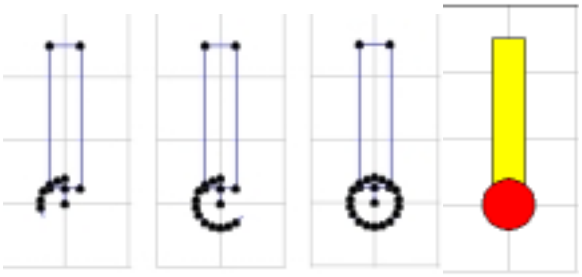
성냥개비의 몸통 타일을 만드는 명령이다. 타일을 만들기 위해 말뚝을 박고 노란색의 타일을 만든다.
말뚝을 박는 순서는 다음과 같다. **이 때 타일의 이름은 자동적으로 t_i로 저장된다.**



<성냥개비 몸통의 말뚝 박기>

```
> new;fd 1.5;rt 100;
> repeat 18 {pt; fd 50/36; rt 20;}
> tile A; tile A, red;
```

새로운 타일을 만들기 위해 new 명령을 한다. 성냥머리 부분의 타일을 만드는 명령이다.
특히 fd 1.5;rt 100;repeat 18 {pt; fd 50/36; rt 20;}이 부분은 원을 그리기 위한 말뚝을 박는 과정이다. 과정을 조금 확대해서 보면 다음과 같다.



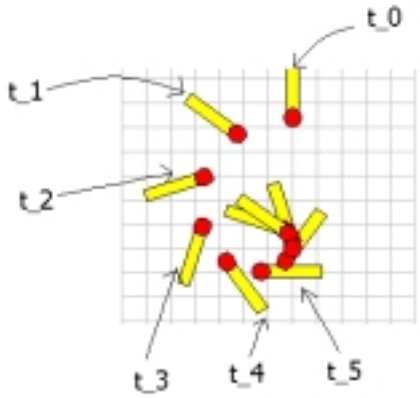
<성냥개비 머리의 말뚝 박기>

```
> for i=0 to 9;
> tile t_i, A;
> next;
```

타일 10개에 대해서 노란 몸통 부분과 (타일 이름 t_i) 빨간 머리부분(타일 이름 A)을 하나로 합치는 명령이다.

```
> for i=0 to 9;
> repeat 18 {tile t_i, fd 3;tile t_i, rt 3*i; pause 30;}
> next;
```

처음 실행시켰을 때 성냥이 발사되는 모양을 나타내는 명령이다.
t_0 라는 타일에 대해서는 3만큼 가고 0 돌고를 18번 반복하는 것이고 t_1 에 대해서는 3만큼 가고 3*1=3
만큼 돌고를 18번 반복하는 것이다.



<성냥 타일의 이동 모습>

특히 t_5 의 경우 도는 각도가 3*5 만큼씩 18번이므로 3*5*18=270 에서x 축과 나란한 방향임을 알 수 있다.

```
> def game() {
> tile t_5, 54.5, -42.5, 360 ;
> tile t_9, -5, -3.5, 135 ;
> tile t_4, -45.5, 38.5, 315 ;
> tile t_0, -23, -29.5, 53.13 ;
> tile t_7, 3.5, 42.5, 315 ;
```

```
> tile t_1, -19.5, 64, 315 ;
> tile t_2, -42, -4, 53.13 ;
> tile t_6, -21.5, 21, 53.13 ;
> tile t_3, 1.5, 2, 53.13 ;
> tile A, 2.5, 28, 206.57 ;
> tile t_8, 89, -9, 216.87 ;
> tt -98,-55;tt blue;
> write '성냥개비 두 개를 옮겨서 물고기의 방향을 바꾸세요'; }
> button game() { '게임 다시하기'; -80,-65}
```

타일의 위치와 방향을 지정하여 타일을 물고기 모양으로 배열 되도록 했다. 예를 들어 tile t_0, -23, -29.5, 53.13 의 경우는 t_0 타일을 x좌표 -23, y좌표 -29.5 에 해당하는 위치에 53.15 의 각도로 위치하도록 하는 것이다.

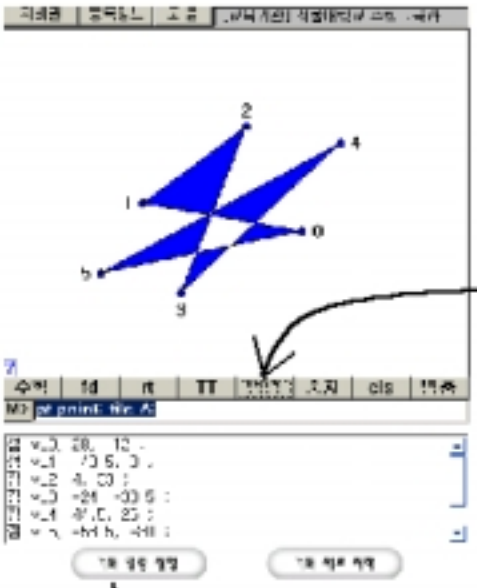
```
> button sol() { '해답 보기'; 40,-65}
> def sol() {
> tile t_1, -19.5, 64, 315 ;
> tile t_3, 1.5, 2, 53.13 ;
> tile t_4, -45.5, 38.5, 315 ;
> tile t_5, 54.5, -42.5, 360 ;
> tile t_6, -21.5, 21, 53.13 ;
> tile t_7, 3.5, 42.5, 315 ;
> tile t_8, 89, -9, 216.87 ;
> tile t_9, -5, -3.5, 135 ;
> tile t_2, 6, 51.5, 53.13 ;
> tile t_0, 25.5, 32.5, 53.13 ;
> tile A, -3.5, 19, 206.57 ;}
```

위 질문에 해당하는 답을 보여준다.

```
> game();
```

실제로 위에서 정의된 game() 함수를 실행한다.

※ 위에서 각 타일의 좌표와 각도를 어떻게 알아낼 수 있었을까?
실제로 이 환경은 위의 그림 프레임과 아래의 명령입력창이 서로 간에 소통을 하고 있기 때문에 그림을 바꾸고 위 프레임의 명령버튼을 누르면 실제 명령을 보여주며, 명령입력창에서 명령을 바꾸고 아래의 거북 명령 실행 버튼을 누르면 위의 그림이 바뀌게 됨을 알 수 있다. 따라서 원하는 그림을 그린 후 명령 버튼을 누르게 되면 거기에 해당하는 명령을 알 수 있게 된다.



위 창에 그림을 그리고 난 후
이 버튼을 누르면
이 그림에 해당하는 명령이
아래의 명령창에 나타난다.

아래 명령입력창에
명령을 입력한 후
이 버튼을 누르면
명령에 해당하는 그림이
위 그림창에 나타난다.