시뮬레이션 기초 및 실습

HW 3

제출일 2019년 6월 02일
과목명 시뮬레이션 기초 및 실습
담당교수 김지범 교수님
학과 산업경영공학과
학번 201401210
이름 강형원

1. Matlab에는 ‘^’을 통하여 x의 n승을 계산할 수 있다. Matlab ‘^’을 사용할 수 없다고 가정하자. 재귀 함수로 x의 n승을 계산하는 방법이 있다. 아래 예는 x^{21} 을 재귀로 계산할 예 이다.

$x^{21} = (x^{10})^2 \cdot x$

→ $x^{10} = (x^5)^2$

→ $x^5 = (x^2)^2 \cdot x$

→ $x^2 = (x)^2$

이 함수의 재귀적 정의는 아래와 같이 정의되어 있다.

$$f(x,n) = \begin{cases} 1 & ,if\ \ n = 0 \\ f(x,\frac{n}{2}) \cdot f(x,\frac{n}{2}) & ,\text{if } n > 0 \text{ and } n\text{이 짝수} \\ f(x,\frac{n-1}{2}) \cdot f(x,\frac{n-1}{2}) \cdot x & ,\text{if } n > 0 \text{ and } n\text{이 홀수} \end{cases}$$

이를 이용하여 아래와 같은 재귀 함수 PowerP(x, n)을 작성해 보자. 어떻게 알고리즘을 작성했는지 설명하자. 작성한 코드가 잘 동작하는지 확인하기 위하여 PowerP(3, 4)를 실행한 출력 결과 (Matlab 명령창)를 캡처하여 보여주자.

```
function y=PowerP(x, n)
% y=x^n, 여기서 n은 0보다 크거나 같은 정수
```

- Algorithm (PowerP.m)

```
function y = PowerP(x, n)

if     % n이 0일 때

elseif     % n이 0보다 크고 짝수 일 때

elseif     % n이 0보다 크고 홀수 일 때

else    % n이 음수일 경우 오류 메시지 출력

end
```

- code (PowerP.m)

```
function y = PowerP(x, n)

if n == 0   % n이 0일 때
    y = 1;
elseif n > 0 && rem(n, 2) == 0  % n이 0보다 크고 짝수 일 때
    y = PowerP(x, n/2) * PowerP(x, n/2);
elseif n > 0 && rem(n, 2) == 1  % n이 0보다 크고 홀수 일 때
    y = PowerP(x, (n-1)/2) * PowerP(x, (n-1)/2) * x;
else    % n이 음수일 경우 오류 메시지 출력
    disp('n은 0보다 크거나 같은 정수이어야 합니다.')
end
```

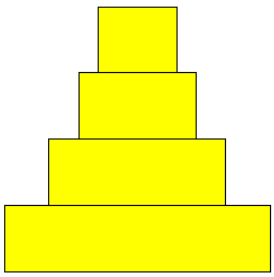
- 결과

```
>> PowerP(3, 4)

ans =

    81
```

2. 아래 왼쪽에 정의된 'DrawRect.m' 함수는 가로 길이 L, 세로 길이 W인 직사각형을 색 c를 이용하여 그리는 함수이다. 이 직사각형의 왼쪽 아래 코너 점의 (x, y)좌표는 (a, b)이다. 예를 들어 DrawRect(2, 1, 2, 2, 'y')는 직사각형의 왼쪽 코너 아래 점이 (x, y) = (2, 1)이고 가로 길이=2, 세로 길이=2이며 노란색 직사각형을 그린다.

'DrawRect' 함수를 반복 호출하고 while 문을 이용하여 아래 오른쪽과 같은 피라미드 형태를 그리고자 한다. 이 피라미드의 제일 아래 직사각형의 왼쪽 아래 코너 좌표는 (x, y)=(0, 0)이다. 이 피라미드의 제일 아래 직사각형은 가로 길이가 W이고 세로 길이가 H이다. 이 두 W와 H 값들은 'input' 함수를 이용하여 사용자에게 직접 입력 받는 값이다. 단, 사용자가 입력 시 H값은 W값보다 항상 같거나 작은 값이라고 가정하자. 각 피라미드의 직사각형의 높이는 모두 H이고, 피라미드의 제일 아래로부터 한 칸 식 위로 올라갈수록 직사각형의 가로 길이는 바로 아래 직사각형의 가로 길이의 2/3 이다. 단, 피라미드의 가장 위의 직사각형의 가로 길이는 반드시 H보다 작지 않도록 피라미드를 만들고자 한다. 어떻게 알고리즘을 작성했는지 설명하자. W=20, H=1인 경우에 대하여 본인이 작성한 코드를 사용하여 피라미드 형태를 만든 출력 결과를 캡쳐 하여 보여주자. 실행 시 총 몇 개의 직사각형이 보이는가?

```
function DrawRect(a, b, L, W, c)
x = [a a+L a+L a];
y = [b b b+W b+W];
fill(x, y, c)
```



- Algorithm (HW3_2.m)

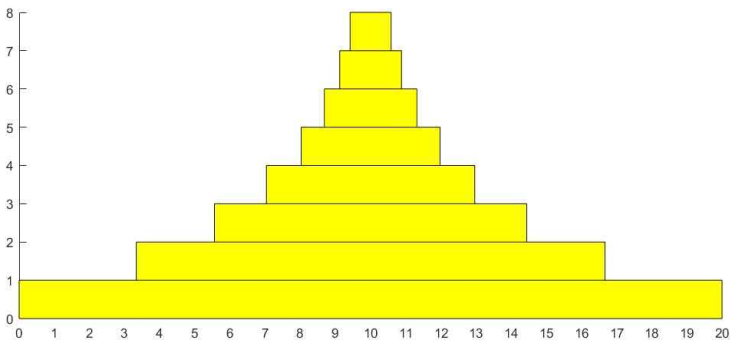
```
x = 0; y = 0;   % x,y를 0으로 초기화

W % 가로 길이 W를 input 함수로 입력 받는다.
H % 세로 길이 H를 input 함수로 입력 받는다.

% 세로 길이 H가 가로 길이 W보다 크면 다시 입력 받는다. (문제에는 없지만 추가로 작성하였습니다.)

% 가로 길이 W가 세로 길이 H보다 작아 질 때까지
% 가로 길이 W를 2/3으로 줄여가며 직사각형을 그린다.
while W > H
```

```
    % 직사각형을 그린다.
    % x 좌표의 위치를 재설정한다.
    % y 좌표의 위치를 재설정한다.
    % 가로 길이 W를 2/3로 줄인다.
end
```

- code (HW3_2.m)

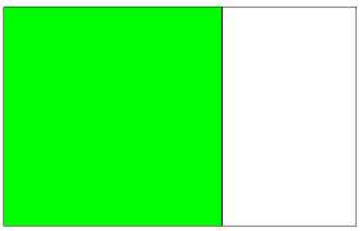
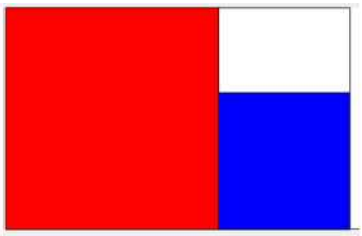
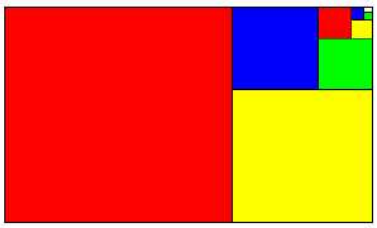
```
x = 0; y = 0;   % x,y를 0으로 초기화

W = input('가로 길이를 입력해주세요.');    % 가로 길이 W를 input 함수로 입력 받는다.
H = input('세로 길이를 입력해주세요.');    % 세로 길이 H를 input 함수로 입력 받는다.

% 세로 길이 H가 가로 길이 W보다 크면 다시 입력 받는다. (문제에는 없지만 추가로 작성하였습니다.)
while W < H
    H = input('세로 길이가 가로의 길이보다 작아야합니다. 세로 길이를 다시 입력해주세요.');
end

figure;

% 가로 길이 W가 세로 길이 H보다 작아 질 때까지
% 가로 길이 W를 2/3으로 줄여가며 직사각형을 그린다.
while W > H
    hold on;
    DrawRect(x, y, W, H, 'y')   % 직사각형을 그린다.
    x = x + W / 6;  % x 좌표의 위치를 재설정한다.
    y = y + H;      % y 좌표의 위치를 재설정한다.
    W = W * 2/3;    % 가로 길이 W를 2/3로 줄인다.
end
```

- 결과



총 8개의 직사각형이 나타난다.

3. 어떤 직사각형에 subdivision 과정을 한 번 수행하면 아래 왼쪽 그림처럼 정사각형과 직사각형으로 나뉘어질 수 있다. 이 subdivision 과정을 한 번 더 수행하면 남은 직사각형 부분을 아래 가운데 그림처럼 다시 정사각형과 직사각형으로 나눌 수 있다. 이를 계속 반복하여 남은 직사각형 부분을 정사각형과 직사각형으로 나눌 수 있다.

이를 수행하는 재귀 함수 'PartitionRect.m'을 divide-and-conquer 방법으로 작성하고자 한다. 이 함수의 원형은 아래 왼쪽과 같이 정의되어 있다. 이 함수는 위에서 설명한 직사각형의 subdivision 과정을 L번 수행하는 함수로 직사각형의 왼쪽 아래 코너 점의 좌표는 (a, b)이고 직사각형의 base (밑변, 가로)와 height (높이, 세로)가 입력인 함수이다. 위의 왼쪽 그림은 L=1, 위의 가운데 그림은 L=2, 위 오른쪽 그림은 L=8인ㄱ 경우의 실행 예이다. subdivision 수행 시 생성되는 정사각형 부분의 색은 무작위로 선택하는데 'r', 'g', 'b', 'y'가 같은 확률로 선택하도록 하고 subdivision 수행 시 남은 직사각형 부분은 흰색 ('w')이 되도록 하자. 실험 시 직사각형의 base=1.6180, height=1.0, a=0, b=0으로 설정하자. 아래 오른쪽에 있는 main 함수를 실행한 결과를 출력해보고 실행 결과 (그림: 총 5개)를 레포트에 출력하자. 어떻게 알고리즘을 작성했는지 리포트에 설명해보자.

Hint: divide and conquer 방법으로 코드를 작성 시 한 가지 방법은 재귀 단계를 작성 시 base가 height보다 큰 경우, base가 height보다 작거나 같은 경우를 모두 생각해 보는 것이다.

PartitionRect.m

```
function PartitionRect(a, b, base, height, L)
if L==0    % 기본 단계
  ?
else       % 재귀 단계
  ?
end
```

main.m

```
clear;
close all;
a=0;
b=0;
base=1.6180;
height=1;
for L=1:5
    figure;
    hold on;
    PartitionRect(a,b,base,height,L)
end
```

- Algorithm (PartitionRect.m)

```
function PartitionRect(a, b, base, height, L)
% 직사각형의 x좌표 선언
% 직사각형의 y좌표 선언

if L==0     % 기본 단계
   % 남은 직사각형 부분을 흰색으로 표시
```

```
else    % 재귀 단계
    % 난수 r 생성
    % 난수에 따라 'r', 'g', 'b', 'y' 중에 무작위로 선택

    if base > height    % base가 height보다 큰 경우
        % 각 변의 길이가 height인 정사각형을 무작위로 선택한 색으로 나타낸다.
        % 나머지 직사각형 부분을 재귀함수로 다시 실행
    else    % height가 base보다 큰 경우
        % 각 변의 길이가 base인 정사각형을 무작위로 선택한 색으로 나타낸다.
        % 나머지 직사각형 부분을 재귀함수로 다시 실행
    end
end
```

- code (PartitionRect.m)

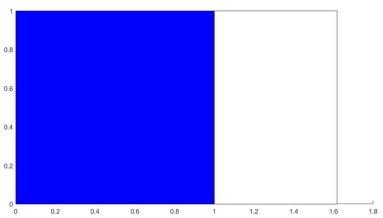
```
function PartitionRect(a, b, base, height, L)
x = [a a+base a+base a];    % 직사각형의 x좌표 선언
y = [b b b+height b+height];    % 직사각형의 y좌표 선언

if L==0    % 기본 단계
    fill(x, y, 'w');    % 남은 직사각형 부분을 흰색으로 표시

else    % 재귀 단계
    r = rand;   % 난수 r 생성
    % 난수에 따라 'r', 'g', 'b', 'y' 중에 무작위로 선택
    if r <= 1/4
        c = 'r';
    elseif r>1/4 && r<=2/4
        c = 'g';
    elseif r>2/4 && r<=3/4
        c = 'b';
    else
        c = 'y';
    end

    if base > height    % base가 height보다 큰 경우
        % 각 변의 길이가 height인 정사각형을 무작위로 선택한 색으로 나타낸다.
        x1 = [a a+height a+height a];
        y1 = [b b b+height b+height];
        fill(x1, y1, c);
        PartitionRect(a+height, b, base-height, height, L-1);   % 나머지 직사각
형 부분을 재귀함수로 다시 실행
    else    % height가 base보다 큰 경우
        % 각 변의 길이가 base인 정사각형을 무작위로 선택한 색으로 나타낸다.
        x1 = [a a+base a+base a];
```

```
        y1 = [b b b+base b+base];
        fill(x1, y1, c);
        PartitionRect(a, b+base, base, height-base, L-1); % 나머지 직사각형 부분
을 재귀함수로 다시 실행
    end
end
```

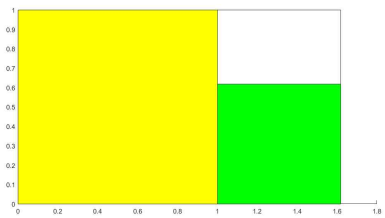
- **설명**

base가 height보다 큰 경우에 height를 네 변의 길이로 하는 정사각형을 네 가지 색상 중에 무작위로 왼쪽에 나타낸 다음 오른쪽 남은 직사각형을 다시 재귀함수를 통해 표현을 하게 되고 그러면 height가 base보다 크게 되므로 이 경우에는 base를 네 변의 길이로 하는 정사각형을 네 가지 색상 중에 무작위로 남은 직사각형의 아래쪽에 나타내고 나머지를 다시 재귀 함수를 통해 이를 반복 작업을 한다.

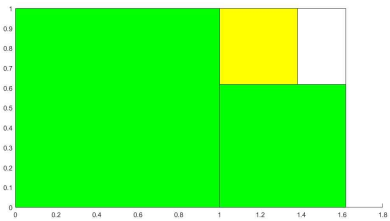
- **결과**



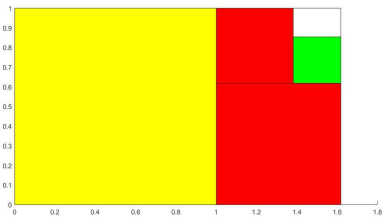
L=1



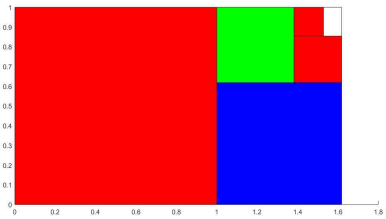
L=2



L=3



L=4



L=5