Snake Game의 이해와 변형 보고서

12조 오예준, 김기현, 변상원, 전성민

1. Snake Game 분석

텍스트이(가) 표시된 사진

자동 생성된 설명function setup()

setup() 함수는 프로그램 실행시 단 한번 호출된다. setup 함수에서는 배경, 화면 크기 등의 환경요소를 정의한다. 또한 framerate()함수를 사용하여 화면에 나타날 프레임 수를 10으로 설정하였다.

function draw()

텍스트이(가) 표시된 사진

자동 생성된 설명

draw() : 프로그램 실행이 중단되거나 noLoop() 함수가 호출되기 전까지 블록 내에 포함된 코드들을 계속 실행.

if (s.eat(food)) : 조건문을 사용하여 뱀이 먹이를 먹었을 경우 pickLocation() 함수를 호출하여 먹이가 생성.

rect() : 사각형의 먹이의 위치와 크기를 설정. 먹이의 위치는 food() 함수의 x, y 값을 불러오고 크기는 scl.

function pickLocation

텍스트이(가) 표시된 사진

자동 생성된 설명

pickLocation() : createVector를 사용하여 food를 정의.

createVector() : 새로운 vector 생성. food는 x와 y성분을 가지고 있는 2차원 vector. 이때 x 성분과 y성분을 random함수와 floor 함수로 설정.

random() : 먹이의 위치가 랜덤하게 생성.

floor() : random(cols)의 값과 random(rows)의 값이 소수점 값이 나올 경우 뱀의 위치와 먹이의 위치가 어긋날 수 있기 때문에 그 위치를 보정하기 위해서 floor함수를 사용하여 소수점을 없앰으로써 먹이의 위치를 보정.

for 반복문을 이용하여서 먹이와 뱀의 거리인 dist를 변수 d로 선언을 하여서 뱀과 먹이의 거리가 1보다 작아지면 먹이가 생성되도록 설정.

function scoreboard()

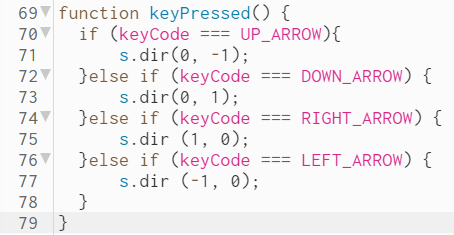
텍스트이(가) 표시된 사진

자동 생성된 설명

function scoreboard()

스코어 보드 - 제목, 점수, 최고점수 출력

function keyPressed()

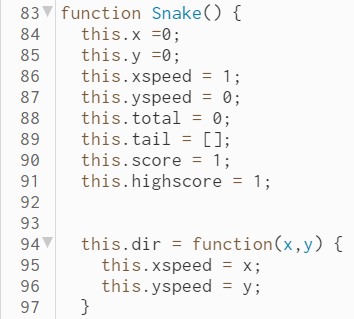


function keyPressed()

사용자 입력 화살표 방향키

뱀 객체의 dir함수를 호출하여 xspeed, yspeed 수정

function Snake()



function Snake()

x: 현재 뱀의 x좌표

y: 현재 뱀의 y좌포

xspeed: 뱀의 x좌표 이동 (초기: 1)

yspeed: 뱀의 y좌표 이동 (초기: 0)

total: 꼬리의 총 개수

tail: 꼬리 백터를 저장하는 배열 ([0]:꼬리의 끝, [length-1]: 꼬리 처음)

score: 현재 점수 (초기: 1)

highscore: 최고점수 (초기: 1)

dir(x, y): 유저가 누른 방향키에 따라 뱀의 이동방향 지정 함수

eat(pos): pos(백터)를 이용해 먹이를 먹었는지 안먹었는지 확인 함수

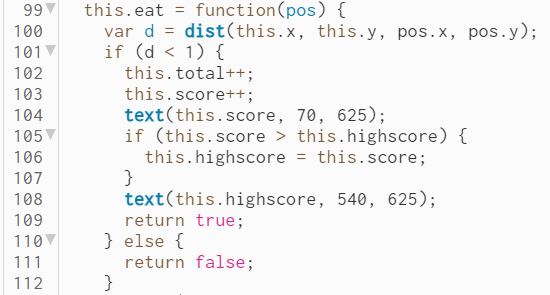
death(): 뱀이 꼬리나 벽에 닿았을 시 점수 초기화 함수

update(): 뱀이 이동 작업 함수

show(): 뱀 출력 함수

this.eat = function(pos)

- 뱀이 먹이를 먹엇을때, 꼬리가 추가되고 점수가 올라가는 함수 구현



this.eat = function(pos) : 뱀 먹이 함수

var d = dist(this.x, this.y, pos.x, pos.y); : 뱀과 먹이 사이 거리 d값

if (d < 1) : 뱀이 먹이를 먹었으면

this.total++; : 뱀 꼬리 추가 +1

this.score++; : 점수 +1

text(this.score, 70, 625); : 점수 표시

if (this.score > this.highscore) { : 점수가 최고점수보다 높으면

this.highscore = this.score; : 최고 점수 업데이트

text(this.highscore, 540, 625) : 최고 점수 표시

return true; : 먹었으면 true

} else {

return false; : 안먹었으면 false

this.death = function() 뱀이 죽었는지 확인 함수

텍스트이(가) 표시된 사진

자동 생성된 설명

this.death = function() { : 뱀이 죽었는지 확인 함수

for (var i = 0; i < this.tail.length; i++) { : i는 뱀의 꼬리 길이까지 반복

var pos = this.tail[i]; : 꼬리 pos에 정의

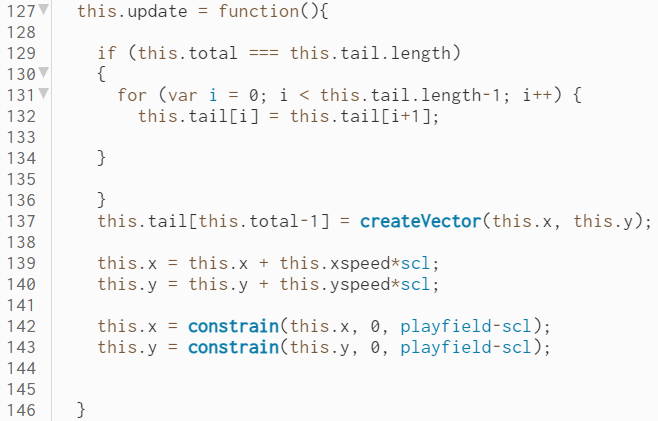
var d = dist(this.x, this.y, pos.x, pos.y); : 뱀과 꼬리 사이 거리 d값

if (d < 1) { : 머리가 꼬리 벽에 닿앗을시 점수와 꼬리 초기화

this.total = 0; : 총 점수 초기화

this.score = 0; : 초기엔 1인데 죽으면 0으로 세팅해둠

this.update = function() **- 뱀 이동을 수시로 업데이트 시키면서 움직임**



this.update = function(){ : 뱀이동 함수

if (this.total === this.tail.length) { : 뱀의 길이와 꼬리 길이가 같을 시

for (var i = 0; i < this.tail.length-1; i++) {

this.tail[i] = this.tail[i+1]; : 배열 인덱스(백터)를 왼쪽으로 한칸씩 이동

this.tail[this.total-1] = createVector(this.x, this.y); : 머리와 가장 가까운 꼬리 업데이트

이동한 만큼 머리 이동

this.x = this.x + this.xspeed\*scl; : 뱀의 속도

this.y = this.y + this.yspeed\*scl; : 뱀의 속도

this.x = constrain(this.x, 0, playfield-scl); : 뱀 x위치 0~580제한

this.y = constrain(this.y, 0, playfield-scl); : 뱀 y위치 0~580제한

this.show = function() **- 뱀과 길이 나타내기**

텍스트이(가) 표시된 사진

자동 생성된 설명

this.show = function(){ : 뱀 출력 함수

for (var i = 0; i < this.tail.length; i++) {

rect(this.tail[i].x, this.tail[i].y, scl, scl); : 뱀 길이x, y좌표 사각형

}

rect(this.x, this.y, scl, scl); : 뱀 x, y좌표 사각형

2. 게임의 변형

시작 화면과 종료 화면의 추가

시작화면을 추가함으로써 프로그램을 시작하자마자 게임이 시작되어 게임 오버되는 경우를 방지하였습니다. 또한 종료 화면을 추가하여서 뱀이 죽었을 경우 게임이 오버되었다는 것을 사용자에게 알려주고 점수를 알려주어서 게임이 리셋 되어 점수를 알 수 없는 경우를 방지하였습니다.

-추가된텍스트이(가) 표시된 사진

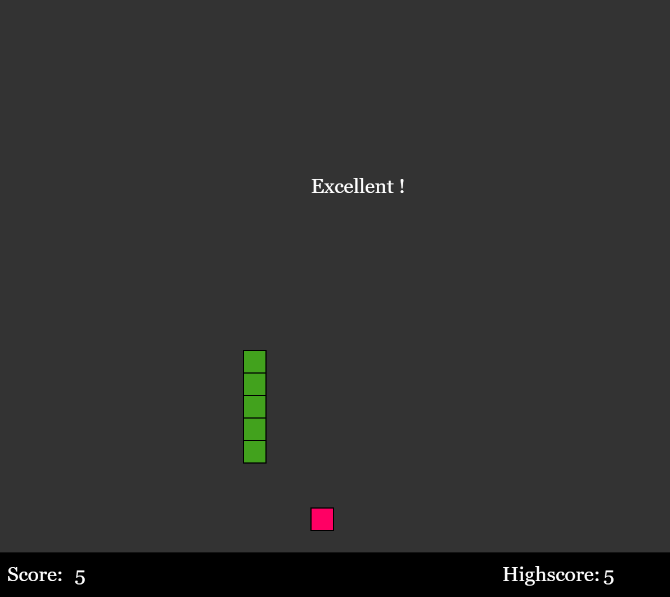
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명 코드

텍스트이(가) 표시된 사진

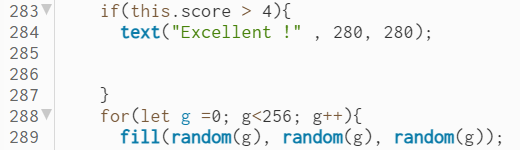
자동 생성된 설명

뱀의 색 변경 및 텍스트 삽입



게임을 진행하는 동안 뱀의 색이 계속 변화하도록 하였습니다. 또한 Score가 4점을 넘으면 Excellent ! 라는 텍스트가 출력되도록 하여서 게임 이용자를 응원할 수 있도록 하였습니다.

-추가된 코드

****